



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

PLATAFORMA DE DESARROLLO DE APLICACIONES EN EL DCC BASADA EN
TÉCNICAS DE DEVOPS

TESIS PARA OPTAR AL GRADO DE MAGISTER EN
TECNOLOGÍAS DE LA INFORMACIÓN

JORGE HERNÁN RETAMAL VALENZUELA

PROFESOR GUÍA:

DANIEL PEROVICH GEROSA

MIEMBROS DE LA COMISIÓN:

NELSON BALOIAN TATARYAN

CECILIA BASTARRICA PIÑEYRO

CARLA VAIRETTI MARTIN

SANTIAGO DE CHILE

2019

Resumen

El desarrollo de aplicaciones en las organizaciones es un aspecto fundamental para el apoyo de la operación y en ese sentido el DCC no es la excepción. En el DCC las aplicaciones son provistas por la facultad o bien son desarrolladas internamente apoyados en estudiantes, memoristas, tesis y académicos. Los desarrollos han sido canalizados a través de un académico quién históricamente se ha hecho cargo de este proceso. Sin embargo, y con el fin de ordenar el proceso y potenciar los resultados, en el 2017 se crea el Área Aplicaciones.

A través de un estudio, la recién creada Área Aplicaciones realizó un levantamiento de todas aquellas aplicaciones desarrolladas internamente y que apoyan la operación. Sin embargo, al ir un poco más allá y consultar acerca del código fuente de las aplicaciones, la documentación o los procedimientos de gestión de incidencias y proyectos, no se encontró información documentada por lo que se observa que no hay gobernanza en el desarrollo y operación de las aplicaciones del DCC.

Para resolver esta problemática, en este trabajo de tesis se crea una plataforma de desarrollo de software basada en el paradigma DevOps. Los principios y alcance de esta plataforma respetan la forma y fuerza de trabajo del DCC. La plataforma está dividida en dos grandes áreas. La primera es un pipeline el cual representa el ciclo de vida de desarrollo de software para las aplicaciones del DCC, abarcando desde tareas de recolección de ideas e incidencias hasta la instalación de las aplicaciones desarrolladas en el ambiente de producción. La segunda es la creación de una plataforma de herramientas concretas que apoyan el uso del pipeline. Estas herramientas asisten al pipeline en todas sus fases, desde la organización y gestión de ideas e incidencias hasta la automatización de la instalación de los componentes en distintos ambientes dependiendo de la fase en la cual se encuentre el desarrollo.

La validación del trabajo se realizó en dos fases. La primera es a través de un piloto en el cual el Área Aplicaciones utiliza el pipeline y las herramientas de apoyo para el desarrollo de una aplicación, para luego solicitar las opiniones de los actores a través de una técnica de retrospectiva. La segunda fase consiste en consultar a las áreas involucradas (Aplicaciones y Sistemas) sus opiniones y observaciones en cuanto a la plataforma definida.

Agradecimientos

A mi hija **Daniela** y mi señora **Jeannette**, por su paciencia durante este proceso, desde soportar mis catarsis hasta los fines de semana y las tardes robadas.

A **mis padres** y a **mi hermana**, por su apoyo permanente, su constante preocupación y por estar ahí siempre que los necesito.

A **Germán Espinoza** y **Pedro Becker**, por su ayuda y por ser parte de este proceso.

A **Daniel Perovich**, mi profesor guía, por su persistencia, energía y entrega y por tantos almuerzos que arruiné al menos una vez a la semana.

Y finalmente a **mis compañeros** en los distintos años en los que fui parte de la universidad.

Tabla de Contenido

1.	Introducción	1
1.1.	Contexto.....	1
1.2.	Problema.....	3
1.3.	Solución.....	4
1.4.	Objetivos	5
1.5.	Metodología.....	6
1.6.	Estructura del informe	7
2.	Marco teórico.....	8
2.1.	Desarrollo y operación tradicional.....	8
2.2.	Automatización	9
2.2.1.	Integración continua	10
2.2.2.	Entrega continua	11
2.2.3.	Implantación continua	12
2.2.4.	Monitoreo continuo.....	13
2.3.	DevOps.....	13
2.3.1.	Principios y cultura DevOps	14
2.3.2.	Métodos de adopción de DevOps.....	16
2.4.	Herramientas del ecosistema DevOps.....	20
3.	Conceptualización de la solución.....	21
3.1.	Análisis del desarrollo en el DCC.....	21
3.1.1.	Aplicaciones tecnológicas	21
3.1.2.	Proceso de desarrollo	25
3.2.	Entrevistas a los interesados.....	25
3.2.1.	Protocolo.....	25
3.2.2.	Ejecución	27
3.2.3.	Resultados.....	27
3.2.4.	Análisis de la situación actual	29
3.3.	Alcance y expectativas	30

3.3.1.	Objetivos de negocio y motivadores.....	30
3.3.2.	Alcance	30
3.3.3.	Interesados.....	34
4.	Proceso y Plataforma de Desarrollo	36
4.1.	Proceso de desarrollo	36
4.1.1.	Pipeline del DCC.....	36
4.1.2.	Roles, personas o equipos que interactúan con el pipeline	39
4.1.3.	Artefactos.....	40
4.2.	Plataforma tecnológica	47
4.2.1.	Tipos de herramientas de apoyo	48
4.2.2.	Proceso de selección de herramientas	49
4.2.3.	Selección de herramientas.....	54
4.2.4.	Ambientes de ejecución.....	59
4.3.	Cobertura de objetivos	60
5.	Validación.....	66
5.1.	Piloto	66
5.1.1.	Plataforma de Apoyo	66
5.1.2.	Modelo de gestión	71
5.1.3.	Ejecución del proyecto.....	74
5.1.4.	Retrospectiva	79
5.2.	Validación por parte de los interesados	83
5.2.1.	Protocolo.....	83
5.2.2.	Ejecución	85
5.2.3.	Resultados.....	86
5.2.4.	Análisis de los resultados	88
6.	Conclusiones	89
6.1.	Trabajo realizado	89
6.2.	Impacto de la solución	90
6.3.	Lecciones aprendidas.....	90

6.4. Trabajo futuro	91
7. Bibliografía	92

Índice de figuras

Figura 1. Áreas principales del DCC.	1
Figura 2. Áreas de apoyo del DCC.	2
Figura 3. Modelo de operación a alto nivel.	5
Figura 4. Trabajo en silos en el de desarrollo y operación de un sistema de software.	8
Figura 5. Niveles de automatización y cómo se relacionan con las etapas del desarrollo de software.	10
Figura 6. Entrega continua representado en diagrama de secuencia.	11
Figura 7. Cadena de entrega.	12
Figura 8. Entrega Agil Disciplinada (Disciplined Agile Delivery) en notación de cadena de valor de Porter [7].	15
Figura 9. Cadena de valor de DevOps en notación de Porter [7].	15
Figura 10. Arquitectura de referencia DevOps [5].	16
Figura 11. Releases rápidos para un retroalimentación de negocio rápido [10].	17
Figura 12 Representación idealista y simplista del proceso [2].	18
Figura 13. Mirada desde el punto de vista de operaciones [5].	19
Figura 14. Aplicaciones del ecosistema DevOps.	20
Figura 15. Pipeline propuesto para el DCC.	37
Figura 16. Utilización del pipeline por lo proyectos.	38
Figura 17. Utilización propia del pipeline por Ingeniería de Software II.	39
Figura 18. Tablero Kanban de ideas e incidencias.	41
Figura 19. Tablero Kanban de proyectos y hotfixes.	43
Figura 20. Manejo de branches en gitflow workflow [16].	45
Figura 21. Tareas ejecutadas por script de instalación.	47
Figura 22. Herramientas de apoyo al pipeline.	48
Figura 23. Pipeline con herramientas concretas.	59
Figura 24. Ambientes de apoyo al pipeline.	59
Figura 25. Tablero de gestión de ideas e incidencias en Trello.	67
Figura 26. Plantilla de tablero de gestión de proyecto en Trello.	67
Figura 27. Configuración inicial de Bitbucket.	68

Figura 28. Wiki en Bitbucket.	69
Figura 29. Repositorio de artefactos Nexus.....	69
Figura 30. Herramienta de automatización de la instalación.	70
Figura 31. Plataforma implementada para el piloto.....	71
Figura 32. Registro de la idea de la evolución de u-comunidad en Trello.	72
Figura 33. Idea en etapa de Diagnose del tablero.	72
Figura 34. Wiki del proyecto U-Comunidad.....	73
Figura 35. Nuevas tareas como resultado del diagnóstico.	73
Figura 36. Tarjeta representativa del proyecto en el tablero Ideas e Incidencias.....	74
Figura 37. Tablero inicial del proyecto.....	75
Figura 38. Espacio de versionado de fuentes para el proyecto en Bitbucket.....	75
Figura 39. Tablero propio del proyecto.	76
Figura 40. Evolución del versionado de fuentes.	76
Figura 41. Tablero de proyecto en ejecución.	77
Figura 42. Ejecución de script de instalación.....	78
Figura 43. Aplicación instalada en ambiente.....	78
Figura 44. Servicio de gestión organizacional en estado Done con etiqueta Complete.....	79

Índice de tablas

Tabla 1. Estado de aplicaciones en el DCC.....	22
Tabla 2. Aplicaciones y tecnologías.....	24
Tabla 3. Entrevista a Interesados en el proceso de desarrollo.	27
Tabla 4. Estructura de contenido entrevistas al DCC.....	51
Tabla 5. Respuestas a la pregunta (a).	52
Tabla 6. Repositorio de artefactos.....	54
Tabla 7. Herramientas de control y versionado de código fuente.	55
Tabla 8. Herramientas de compilación.	56
Tabla 9. Herramientas de monitoreo de logs.	56
Tabla 10. Plataforma de contenedores.....	57
Tabla 11. Gestión de ideas e incidencias.	58
Tabla 12. Verificación de completitud de alcance y expectativas.	65
Tabla 13. Protocolo reunión de retrospectiva.	80
Tabla 14. Respuestas obtenidas en la retrospectiva.....	82
Tabla 15. Estructura de contenido presentación del pipeline.....	85
Tabla 16. Respuestas situación anterior vs situación actual.....	86
Tabla 17. Respuestas logro de principios.....	86
Tabla 18. Calificación de la plataforma en términos del rol.	87
Tabla 19. Calificación de la plataforma en términos del DCC.....	87

1. Introducción

1.1. Contexto

El Departamento de Ciencias de la Computación (DCC) de la Universidad de Chile nace en 1975, fundado por un grupo de académicos conscientes de las nuevas tecnologías que se estaban desarrollando. El día de hoy el DCC es uno de los mayores referentes en investigación en Ciencias de la Computación en nuestro país, como también en educación continua, y carreras de pre- y postgrado. El Departamento lleva adelante su operación utilizando un conjunto de aplicaciones que asisten a sus procesos y actividades diarias. Dada la velocidad con la cual las tecnologías de la información se involucran cada vez más en el día a día y evolucionan, es que se hace necesario tener un control sobre las aplicaciones que brindan apoyo a la operación, desde su desarrollo e integración hasta la puesta en producción.

Dentro de la organización del DCC se pueden identificar cuatro áreas principales: Pregrado, Postgrado, Extensión e Investigación, las cuales se subdividen en sub-áreas según se muestra en la Figura 1.

Prácticas profesionales	Proyecto de Software	Titulación	MTI	MSc
Carrera	Pregrado		Postgrado	Doctorado
Primer año				
Vinculación externa	Extensión		Investigación	Post doctorado
PEC				Proyectos externos
				Redes académicas

MTI: Magíster en Tecnologías de la Información

MSc: Magíster en Ciencias

PEC: Programa Educación Continua

Figura 1. Áreas principales del DCC.

A su vez, también existen dos áreas de apoyo transversales a las anteriormente mencionadas: Gestión y Servicios. Al igual que las áreas principales, éstas se encuentran subdivididas como se muestra en la Figura 2.



Figura 2. Áreas de apoyo del DCC.

En la actualidad, todas las áreas y sub-áreas del DCC manejan sus aplicaciones de manera separada, y por lo tanto la información que éstas gestionan se encuentra distribuida y aislada. Esto provoca entonces que eventualmente haya información repetida o inconsistente entre una aplicación y otra. Surge entonces una necesidad de interoperabilidad tanto de la información como entre las diversas aplicaciones del departamento.

El área encargada de la gestión de la plataforma tecnológica y de mantener operativas las aplicaciones ha sido Sistemas. Históricamente, en el DCC no ha existido un área encargada de la construcción de las aplicaciones. Las aplicaciones existentes han sido el resultado de iniciativas individuales de algunos académicos del departamento, y se han construido, mantenido y evolucionado mediante el esfuerzo de estudiantes en contextos como: el curso de pregrado Ingeniería de Software II, memorias de ingeniería, proyectos de grado de Magíster en Tecnologías de la Información, y el desarrollo in-house realizado por recién egresados o por los académicos propiamente. Estos esfuerzos de desarrollo han sido inorgánicos y no se han gestionado en forma centralizada, lo cual ha traído consecuencias serias: (i) no se tiene registro de qué aplicaciones están operativas y en uso, (ii) no se registra quién o quiénes han sido los desarrolladores de cada aplicación, (iii) no se lleva registro de las diferentes versiones de cada aplicación, (iv) no se cuenta con la documentación o el código fuente de cada aplicación ya que generalmente está en poder del último desarrollador, y (v) no se sabe si el código fuente con el que se cuenta corresponde a la versión que está en operación.

Recientemente, el DCC ha creado el área de soporte Aplicaciones cuyas responsabilidades principales son gestionar los esfuerzos de desarrollo de las aplicaciones del departamento, llevar registro y control de los artefactos involucrados en cada aplicación (documentación, código fuente, archivos de instalación, etc.), y llevar registro de incidencias y nuevas necesidades de cada aplicación. En el corto y mediano plazo, el

Área Aplicaciones será responsable del desarrollo en sí mismo, coordinando el esfuerzo de estudiantes y académicos (fuerza de desarrollo actual) y aportando a su vez como área de desarrollo de software con recursos propios. Adicionalmente, esta nueva área será responsable de coordinar los esfuerzos con el área de Sistemas, de forma de facilitar la puesta en producción de cada nueva aplicación o la actualización en cuanto a versión de una aplicación, y la recolección de retroalimentación de la ejecución de las aplicaciones (por ejemplo, la información en los logs).

Los principales desafíos que tiene esta nueva área son:

- El relevamiento y la recolección de los artefactos correspondientes a todas las aplicaciones en desarrollo y en operación.
- La gestión de las aplicaciones, sus artefactos, sus incidencias y sus nuevas necesidades.
- La promoción de la integración de las aplicaciones.
- La definición de políticas que regulen el desarrollo, pruebas, la puesta en producción, y la recolección de retroalimentación de la ejecución (logs).
- La definición y puesta en producción de las herramientas necesarias para asistir y automatizar la gestión, el desarrollo y la puesta en producción de las aplicaciones.

1.2. Problema

Este proyecto de tesis aborda parte de estos desafíos que enfrenta el Área Aplicaciones del departamento. En particular, resuelve los siguientes problemas:

- *No existe un proceso definido documentado de desarrollo de aplicaciones en el DCC.* Si bien es cierto que, al consultar a los actores internos del departamento involucrados en el ciclo de vida de desarrollo de aplicaciones, hay nociones de un procedimiento tanto de desarrollo como de soporte de aplicaciones, éste no está definido formalmente ni tampoco documentado.
- *No se cuenta con versionado de código fuente.* Actualmente en el DCC no se utiliza un controlador de versiones. Como consecuencia, no se lleva registro de qué versiones hay de cada aplicación, de quiénes han sido los desarrolladores involucrados en construirlas, y cuál de estas versiones corresponde a la que está en producción. A su vez, no se tiene acceso a las versiones recientes del código fuente cuando se requiere realizar un mantenimiento o extensión a una aplicación.
- *Solo se cuenta con el ambiente de producción.* Actualmente, cada desarrollador trabaja en su propio ambiente de trabajo, y no se le provee un ambiente en el cual realizar pruebas de integración o pruebas de aceptación de usuario. Como consecuencia, las aplicaciones fallan al ser puestas en producción, ya sea porque las configuraciones locales (del desarrollador) no corresponden a las del ambiente de producción, o porque las aplicaciones no han sido probadas por usuarios finales hasta que no llegan al ambiente de producción. En los casos en que se utiliza un ambiente específico, distinto al de producción, éste debe ser creado por demanda por el Área Sistemas, sin una gestión centralizada de estos ambientes, de su seguridad y de cuándo deben dejar de estar operativos.

- *Instalación manual por parte de los desarrolladores.* Actualmente, la puesta en producción de las aplicaciones la realizan los desarrolladores directamente. Esto conlleva un problema de seguridad ya que el Área Sistemas debe entregar credenciales al equipo de desarrollo para acceder a los servidores de producción, tanto a ambientes de ejecución (servidores virtuales) como a bases de datos. Como consecuencia, las credenciales de los servidores se hacen públicas al ser compartidas con desarrolladores que estarán en la construcción de aplicaciones por un período corto de tiempo (el semestre o año que lleva el curso en el cual está realizando el desarrollo). Además, es muy común que al entregar solo el ejecutable, las piezas de código conocidas y versionadas no correspondan a las que lo componen.
- *No se monitorean las aplicaciones en operación.* Aunque la mayoría de los servidores y las aplicaciones registran logs de su operación, el análisis de estos logs se realiza únicamente por demanda, usualmente cuando el problema con una aplicación es crítico y bloqueante. Como consecuencia, en el ambiente de producción ocurren problemas con las aplicaciones que no son detectados, o que son detectados cuando el problema se hace crítico solamente, impidiendo que se asignen esfuerzos a la corrección o mejora de las aplicaciones antes de que éstas dejen de estar operativas.
- *No hay registro centralizado de requisitos e incidencias.* El departamento no lleva un registro de qué nuevas necesidades deben incluirse en las aplicaciones, o de qué errores se han encontrado en las aplicaciones y deben ser mejoradas, sino que se recopilan recién al momento en el cual se cuenta con un nuevo recurso para el desarrollo o mantenimiento. Como consecuencia, los requerimientos e incidencias capturados son usualmente menos de los que realmente están presentes en las aplicaciones. A su vez, no hay cómo priorizar estas necesidades y por lo tanto no se analiza a qué aplicación es más conveniente dedicarle el esfuerzo cuando hay nuevos recursos disponibles.

Este trabajo de tesis aborda estos problemas, definiendo el proceso y el conjunto de herramientas que permite resolverlos y asistir así a la gestión y al desarrollo de aplicaciones del departamento. La complejidad del problema radica no solo en la creación de un ecosistema de herramientas integrado que permita la automatización de tareas como la integración y prueba de cada aplicación, su puesta en producción, y la recolección de logs para su análisis, sino también en la creación de un procedimiento de desarrollo de software ajustado tanto técnica como culturalmente a la realidad del DCC.

1.3. Solución

Para resolver el problema planteado, se define un modelo de operación basado en el paradigma DevOps, ajustado al nivel de madurez actual del Área Sistemas y de la nueva Área Aplicaciones del DCC. El modelo está soportado por una arquitectura de referencia para los ambientes, considerando el ciclo de vida de desarrollo de software a nivel técnico, es decir, desde su codificación hasta su puesta en producción. La validación consta de dos fases. La primera es a través de un piloto en el cual el Área Aplicaciones utiliza el pipeline y las herramientas de apoyo para el desarrollo de una aplicación, para luego solicitar las opiniones de los actores a través de una técnica de retrospectiva. La segunda fase consiste en consultar a las áreas involucradas (Aplicaciones y Sistemas) sus opiniones y observaciones en cuanto a la mejora del proceso.

En la Figura 3 se muestra en alto nivel el modelo de operación que se definió. Se parte de un requerimiento o incidente el cual es capturado en un software de tickets. Este ticket llega al equipo que desarrolla el requerimiento o resuelve el incidente, implementando los cambios necesarios. Una vez implementados, se hace commit al controlador de versiones (CV) y el desarrollo va avanzando por las etapas del pipeline, siendo integrado con el resto de las aplicaciones y probado por los usuarios en pruebas de aceptación de usuario. Si estas pruebas son exitosas, el componente pasa a producción.



Figura 3. Modelo de operación a alto nivel.

1.4. Objetivos

El objetivo general de este trabajo es proveer al DCC de un modelo de gestión del desarrollo y operación de aplicaciones, basado en técnicas de DevOps, aumentando la gobernanza de la plataforma y su evolución. Al cumplir este objetivo se busca que, tanto los nuevos desarrollos como los actuales, utilicen esta nueva arquitectura de desarrollo y a nivel técnico evolucionen en ambientes controlados con un versionado de código fuente que permita una trazabilidad efectiva hacia el requerimiento o mantenimiento correctivo que se está aplicando.

Para lograr dicho objetivo general, se plantean los siguientes objetivos específicos:

- Definición de un modelo de gestión que permita la gobernanza de las aplicaciones.
- Definición de la arquitectura tecnológica que asista a la gestión y operación de la plataforma.
- Implementación de la arquitectura con herramientas concretas, alineadas a la cultura organizacional del DCC.

1.5. Metodología

Para lograr los objetivos previamente descritos se utilizó la siguiente metodología de trabajo, organizada en tres fases.

Fase preliminar.

1. *Entrevistas.* Se realizan entrevistas a los distintos involucrados en el desarrollo de software en el DCC con el fin de relevar el procedimiento actual.
2. *Análisis de la situación actual.* Se analizan los resultados de las entrevistas y se detectan posibles puntos de mejora.
3. *Investigar el paradigma DevOps.* En esta actividad se revisa la literatura para capturar los métodos y técnicas vinculados a este paradigma, en particular integración continua, entrega continua y puesta en producción continua.
4. *Contexto de la arquitectura.* Validar en conjunto con el DCC la realidad en la cual está inserta la arquitectura o procedimiento definido y las herramientas de apoyo.

Fase de ejecución.

5. *Definición de la arquitectura.* Entregar al DCC una arquitectura que conste de un proceso de desarrollo basado en DevOps apoyado sobre una base tecnológica.
6. *Definición del proceso (propuesta).* Entregar al DCC un procedimiento de entrega de software refinado para su situación particular.
7. *Investigación de herramientas de apoyo.* Se realizan entrevistas con el fin de obtener información acerca de los criterios de selección de las herramientas de apoyo al proceso propuesto y en base a esto, seleccionar las que cumplan con dichos criterios.

Fase de validación.

8. *Seleccionar la aplicación a utilizar.* Decidir dentro de las aplicaciones actualmente en desarrollo cuál es candidata para utilizar el nuevo modelo de entrega.
9. *Realizar un piloto.* Se presenta la arquitectura y sus herramientas de apoyo al equipo de desarrollo para que construyan la aplicación seleccionada anteriormente utilizando el nuevo modelo de entrega.
10. *Validación con interesados.* Se entrevista a todos los involucrados sus impresiones relacionadas con la plataforma.

1.6. Estructura del informe

Este trabajo de tesis está estructurado de la siguiente forma. En el Capítulo 2 se explican los conceptos sobre los cuales se basa el trabajo de tesis, dando una base para el entendimiento de los capítulos posteriores. En el Capítulo 3 se indaga en el problema a través de investigación propia y de entrevistas, para luego establecer los principios y el alcance de la solución. En el Capítulo 4 se describe la arquitectura en sus distintas dimensiones, éstas son, definición del pipeline y creación de la plataforma con sus herramientas de apoyo al pipeline. En el Capítulo 5 se documenta la validación, la cual involucra dos ámbitos. El primero es una validación a través de la ejecución de un piloto en el cual se realiza un desarrollo utilizando el pipeline y las herramientas de apoyo. El segundo es la validación por parte de los interesados, obteniendo sus impresiones respecto al cambio de paradigma en el desarrollo de aplicaciones en el DCC. El Capítulo 6 presenta las conclusiones y propone posibles trabajos futuros.

2. Marco teórico

En este capítulo se discuten los conceptos y técnicas actuales para la automatización de la puesta en producción de sistemas de software. Estas técnicas promueven un cambio en el proceso de entrega de software apalancado en tecnología, reduciendo así el esfuerzo y los errores que implica el hacerlo manualmente.

2.1. Desarrollo y operación tradicional

Por lo general, un proceso de desarrollo de software incluye una fase de análisis de las necesidades y requisitos, un diseño del software que los resuelve, y la construcción de éste generando un entregable desde el equipo de desarrollo. Este entregable es casi siempre probado por un equipo encargado del aseguramiento de calidad de software (QA) o bien por el usuario en pruebas de aceptación de usuario (UAT). En este punto se valida que las funcionalidades solicitadas se comportan de manera correcta a través de una serie de casos pertenecientes a un plan de pruebas el cual es ejecutado por los miembros del equipo de manera manual.

Finalmente, si las pruebas de aceptación finalizan correctamente, el software es entregado a operaciones, área que toma el software, lo instala en el ambiente en los cuales éste va a ser utilizado (ambiente de producción) y lo mantiene disponible mientras es utilizado por los usuarios. En paralelo, en base a los errores que vayan identificándose en el uso y la operación del sistema, sean estos funcionales o técnicos (performance, migraciones de plataformas, refinamiento de logs, entre otros), el equipo de desarrollo preparará una nueva versión del sistema en base al mismo proceso descrito anteriormente. La Figura 4 grafica esta forma de trabajo en paralelo.

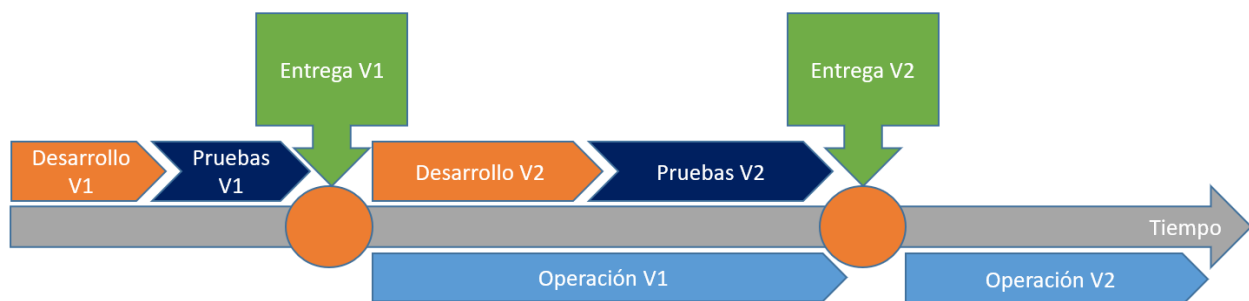


Figura 4. Trabajo en silos en el desarrollo y operación de un sistema de software.

Como se puede notar, esta forma de trabajar divide a los equipos por su “tipo” de trabajo. Esta forma de trabajar es llamada *trabajar en silos*. Los silos crean ambientes de trabajo dispersos e individuales dentro de una organización [1]. El área de desarrollo se especializa en escribir código y por lo tanto no es experta en las pruebas de software. Esta tarea es ejecutada por áreas dedicadas a probar software. Debido a que dejar el software en producción y mantenerlo operativo requiere otras habilidades que no son el

desarrollo ni tampoco de pruebas de software, se crea el área de operaciones. Dividir las áreas de trabajo parece beneficiar también el manejo de la misma. Además del equipo especializado, cada área tiene su propio jefe quien complementa el requerimiento individual necesario para esa área específica [2].

Todo pareciera indicar que esta forma de trabajar funciona sin problemas. Sin embargo, el conflicto entre estas áreas surge, por ejemplo, en la combinación de las siguientes situaciones [2]:

- *Necesidad de cambio*: el área de desarrollo, debido a necesidades de negocio y/o técnicas, produce cambios. Por lo tanto, se requiere que estos cambios sean instalados en producción.
- *Necesidad de estabilidad*: una vez que el software está entregado, el área de operaciones evita hacer cambios al software para asegurar las condiciones estables de los ambientes de producción.

Al estar estas áreas enfocadas en sus objetivos y responsabilidades, cada uno de los participantes en la cadena de la entrega de software actuará intentando resguardar la continuidad de su trabajo y, en el momento de surgir un error, cada área se hará cargo de su alcance dentro de la cadena. A su vez, dudará del trabajo realizado por las otras partes [2] y, peor aún, es que se incrementará la deuda técnica¹ [3].

Con el fin de mejorar este proceso y proponer una respuesta a reducir el error dentro de las entregas de cada área es que desde la tecnología surge la automatización del proceso de entrega de software. Identificar las tareas repetibles y automatizarlas libera a las personas y les permite enfocarse en actividades que producen valor y además reducen la posibilidad de error humano [4].

2.2. Automatización

Como se mencionó en la sección anterior, es posible optimizar las tareas realizadas en las fases del proceso de entrega de software a través de la tecnología. Al realizar esta actividad se elimina la “intervención humana” reduciendo con esto el error humano en el proceso.

En la Figura 5 se muestran las tareas que pueden ser consideradas repetibles en el proceso de entrega de software. Dependiendo del nivel de automatización que tengamos en nuestra organización es la denominación que tendrá este proceso. Por ejemplo, para realizar entregas continuas, debemos estar ejecutando integración continua, así como para realizar implantación continua debemos estar ejecutando entregas continuas y por transitividad, integración continua.

¹ El término “deuda técnica” describe cómo las decisiones que tomamos para resolver problemas inmediatos pueden conducir a problemas que a la larga son más difíciles de arreglar [3].

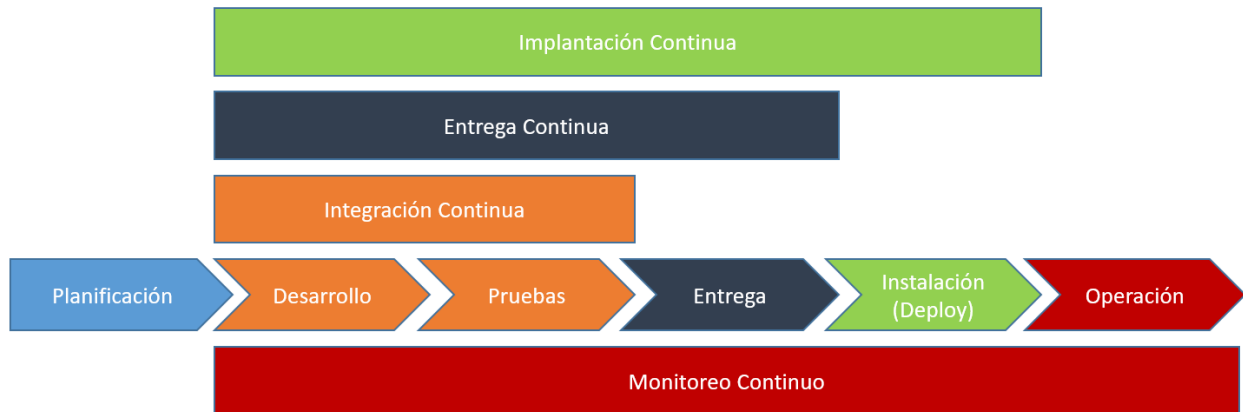


Figura 5. Niveles de automatización y cómo se relacionan con las etapas del desarrollo de software.

2.2.1. Integración continua

El desarrollo de software involucra a un gran número de equipos o personas con responsabilidades transversales, incluyendo dueños de líneas de negocio, analistas de negocio, arquitectos empresariales y de software, desarrolladores, QA, personal de operaciones, especialistas de seguridad, proveedores y partners. Los integrantes de estos equipos trabajan en diversas plataformas y pueden estar en diferentes lugares físicos. El desarrollo colaborativo permite a estos especialistas trabajar en conjunto proveyendo un conjunto de prácticas comunes y una plataforma en común que pueden usar para crear y entregar software [5].

Integración Continua (Continuous Integration en inglés) corresponde a la automatización de la integración de código entre los distintos desarrolladores, asegurando la correcta generación de ejecutables durante la etapa de desarrollo. Además, en las herramientas de integración continua es posible definir la ejecución tanto de pruebas de integración como pruebas unitarias para verificar que las funcionalidades del sistema continúen disponibles luego de la modificación de piezas de código.

Debido a la naturaleza de la integración continua, podemos generar un cambio de paradigma. Sin ella, el software está potencialmente roto hasta que alguien pruebe que funciona, usualmente durante la fase de pruebas o integración. Usando integración continua, el software está probado para funcionar (asumiendo un conjunto de pruebas automatizadas suficientes y comprensivas de su contexto) con cada nuevo cambio, y se sabe en el momento en que se rompe y, por lo tanto, se puede arreglar de forma inmediata. Los equipos que usan integración continua efectivamente están capacitados para proveer software mucho más rápido y con menos errores que los equipos que no tienen esta práctica. Los errores son capturados mucho antes en el proceso de entrega donde son más baratos de corregir, ahorrando significativamente en tiempo y costos. Por lo tanto, es una práctica esencial para equipos profesionales, quizás tan importante como usar un controlador de versiones [6].

Una forma de definir integración continua es tener ejecuciones automáticas entre una fase y la siguiente, luego de las pruebas de integración. Esto es, si la construcción del ejecutable es exitosa, entonces las

pruebas de integración son ejecutadas. Si no, el desarrollador responsable por la falla es notificado [7].

La Integración Continua reduce el riesgo al mantener los fuentes integrados y probados, asegurando la repetitividad y la retroalimentación temprana para los desarrolladores, debido a que ante cualquier problema que se haya encontrado, este será notificado.

2.2.2. Entrega continua

Entrega Continua (Continuous Delivery en inglés) toma el concepto de integración continua al siguiente nivel [5]. Consiste en una disciplina de desarrollo de software donde se construye de tal manera que este puede ser liberado a producción en cualquier momento [8].

Esto se logra integrando continuamente el software hecho por el equipo de desarrollo, construyendo ejecutables y corriendo pruebas tanto automatizadas como manuales para detectar problemas [8]. Si todo finaliza de manera correcta, se obtiene un ejecutable el cual puede ser entregado para ser instalado. Esto se puede ver en la Figura 6 donde se puede notar la interacción entre integración continua y entrega continua. Esta última se considera desde las pruebas automatizadas hasta la liberación del ejecutable.

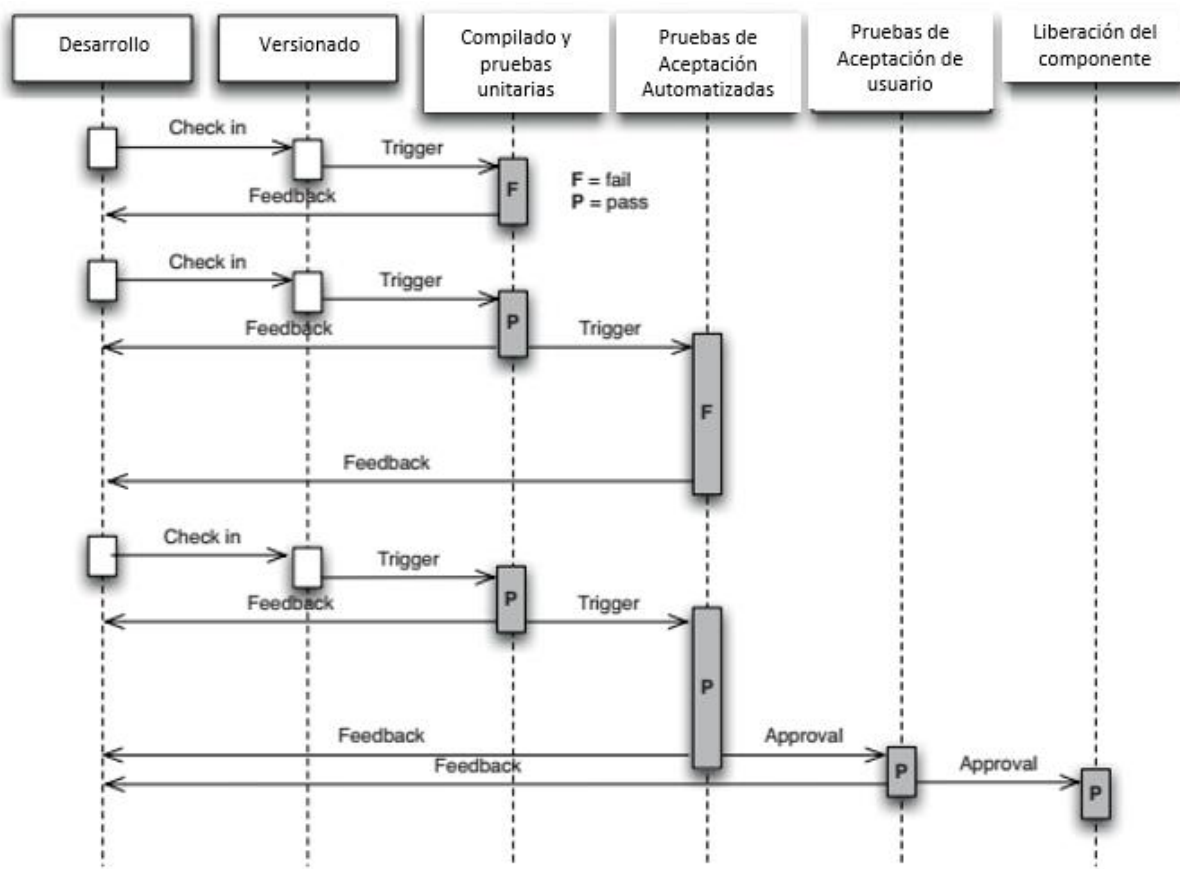


Figura 6. Entrega continua representado en diagrama de secuencia.

2.2.3. Implantación continua

Implantación continua corresponde a la etapa posterior a entrega continua, es decir, debemos haber adoptado este último concepto para alcanzar esta etapa, y con esta práctica llevamos la automatización un paso más allá realizando instalaciones automáticas en los sistemas de producción [7].

Usualmente entrega continua es confundida con implantación continua. Implantación continua significa que cada cambio que viaja a través de la cadena de entrega, automáticamente se deja en producción, lo que resulta potencialmente en muchas implantaciones (deployments) cada día. Por otro lado, entrega continua consiste en realizar entregas, pero es posible elegir instalar o no instalar los ejecutables, usualmente debido a que el negocio prefiere una tasa más lenta de instalaciones, o bien para realizar validaciones adicionales a las automatizadas. De todas maneras, como se mencionó anteriormente, si se quiere hacer implantación continua, debemos estar haciendo entrega continua [4].

Podemos definir implantación continua como un proceso ininterrumpido que implanta el software instantáneamente en ambientes productivos [9]. El conjunto de scripts de implantación automatizado sirve como documentación y deben mantenerse actualizados y completos, de lo contrario la implantación no funcionará [6].

Pieza clave en este proceso es la cadena de entrega (Figura 7). En un nivel abstracto, la cadena de entrega es una versión automatizada del proceso de obtener software desde el controlador de versiones a las manos del usuario (ambiente de producción). Cada cambio en el software se canaliza a través de un proceso complejo en su camino a ser liberado. Este proceso involucra construir el software, seguido de un progreso de estas construcciones a través de múltiples etapas de pruebas e implantaciones [6]. Estas etapas pueden variar de una organización a otra, y también podrían variar de una aplicación a otra dependiendo de las necesidades de la organización, del proceso de desarrollo y entrega de software y la madurez en la que se encuentra el proceso. El nivel de automatización también puede variar; algunas organizaciones automatizan toda la cadena de entrega, otras ponen el software a través de chequeos manuales debido a regulaciones o requerimientos de la compañía [5].

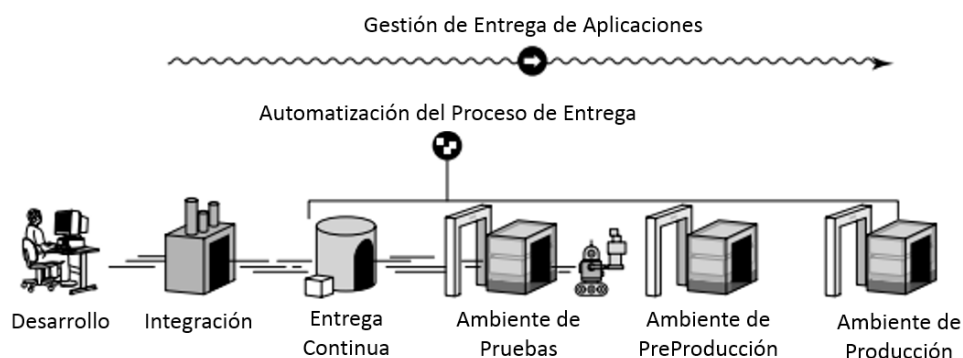


Figura 7. Cadena de entrega.

El resultado final de esto (en términos de metodologías ágiles) es un *pull system*. Los equipos de pruebas esperan una nueva entrega e instalan los ejecutables en la medida que el equipo de desarrollo finaliza. Operaciones puede implantar ejecutables en ambientes de ensayo (staging en inglés) y producción pulsando un botón. Los desarrolladores pueden ver los ejecutables que han ido pasando a través de las etapas del proceso de versión y qué problemas se han ido encontrando. Los jefes de proyecto, interesados, entre otros pueden ver métricas clave en cada ciclo, rendimiento y calidad del código. Como resultado, todos en el proceso de entrega obtienen acceso a las cosas que necesitan cuando las necesitan, y visibilidad dentro del proceso de entrega para mejorar la retroalimentación. De esta manera se pueden identificar los cuellos de botella, optimizarlos y quitarlos. Esto lleva a un proceso de entrega no solo más rápido, sino que también más seguro [6].

2.2.4. Monitoreo continuo

Una buena práctica en el proceso de desarrollo de software es involucrar al área de operaciones lo más temprano que sea posible durante el ciclo de vida de la aplicación en vez de que operaciones haga trabajo manual que venga de tickets de trabajo. Creando una API de autoservicios que posibiliten a los desarrolladores crear ambientes donde puedan probar e instalar código, monitorear y mostrar métricas de producción. Haciendo esto, operaciones se parece más a desarrollo involucrado en la construcción del producto, donde su entregable es la plataforma en la cual los desarrolladores usan para instalar y probar de forma rápida y segura, así como también ejecutar sus servicios en ambientes productivos [3].

2.3. DevOps

DevOps es un conjunto de prácticas destinadas a reducir el tiempo entre la realización de un cambio a un sistema (entendiéndolo como un requerimiento de negocio) y el cambio puesto en producción, mientras se asegura su calidad [7]. Este paradigma incluye todas las características técnicas descritas en la Sección 2.2.

Sin embargo, DevOps va más allá de la tecnología. Es el resultado de aplicar los principios más confiables de un dominio de manufactura y liderazgo a la cadena de valor de TI. DevOps se basa en los conocimientos del paradigma ágil, teoría de restricciones, el sistema de producción de Toyota, ingeniería de resiliencia, aprendizaje organizacional, cultura de seguridad, factor humano, entre otras. DevOps además busca integrar las culturas de gestión de alta confianza, liderazgo de servicio y gestión de cambio organizacional, buscando un resultado tanto de confiabilidad como también de calidad, estabilidad y seguridad a bajo costo y esfuerzo. Provee un flujo acelerado y de confianza a través de la cadena de valor de la tecnología, incluyendo gestión del producto, desarrollo, QA, operaciones TI y seguridad informática [3].

El resultado técnico, arquitectónico y prácticas culturales representan la convergencia de varios movimientos de gestión y filosóficos [3], ya que DevOps engloba numerosas actividades y aspectos como [2]:

- *Cultura*: personas sobre procesos y herramientas. Software está hecho para y por personas.
- *Automatización*: la automatización es esencial para DevOps para obtener una rápida retroalimentación.
- *Métricas*: DevOps a través de sus herramientas, ayuda a obtener métricas. La calidad y los incentivos compartidos (o al menos alineados) son críticos.
- *Compartir*: Crea una cultura donde las personas comparten ideas, procesos y herramientas.

Las bases fundamentales para el éxito de DevOps son la cultura de la confianza y el sentido de comunidad. Todo comienza con cómo las personas se perciben unas a otras. Esto es, cuál es la cultura que tiene la compañía “nosotros contra ellos” o “todos nosotros”. Así, DevOps centra el concepto de “compartir”: compartir ideas, problemas, procesos, herramientas y objetivos [2].

2.3.1. Principios y cultura DevOps

El desarrollo de software ágil ha roto algunos silos entre el análisis de requerimientos, la construcción y las pruebas. Implantación (deployment), operaciones y mantención son otras actividades que han experimentado una separación similar del resto del proceso de desarrollo de software. El movimiento DevOps está dirigido para eliminar estos silos y promover la cooperación entre desarrollo y operaciones [4].

DevOps ha sido posible principalmente debido a la combinación de nuevas herramientas de operaciones y prácticas de ingeniería ágil. Sin embargo, éstas no son suficientes para obtener sus beneficios. Incluso con las mejores herramientas, DevOps es solo otra moda si no se promueve la cultura adecuada [4].

Además, el desarrollo ágil de software considera diferentes roles, incluyendo programadores, testers y personal de aseguramiento de calidad (QA). Estos distintos expertos componen el equipo de desarrollo funcional transversal. Esta “visión de un solo equipo” los acerca más de lo que han estado antes que el movimiento ágil golpeará la industria. El desarrollo de software ágil es ahora corriente. Los principios de los métodos ágiles están enfocados en definir y construir software [2].

Los equipos de desarrollo que siguen un proceso ágil o iterativo típico, como se muestra en la Figura 8, transforman las ideas en historias de usuario y algún tipo de especificación sobre una característica que será implementada en código dentro de la aplicación o servicio que será construido. El código entonces es validado en el repositorio de control de versiones, donde cada cambio es integrado y probado con el resto del sistema [3]. DevOps provee patrones para fomentar la colaboración a los interesados en el proyecto y usa procesos, así como herramientas para agilizar el proceso de entrega de software y reducir el tiempo total del ciclo [2].

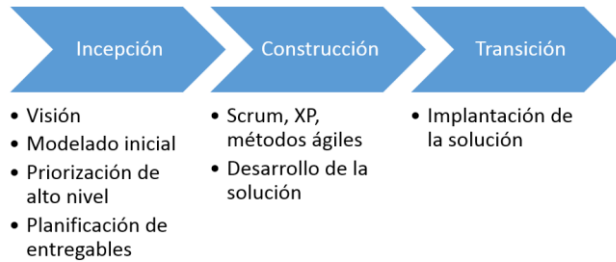


Figura 8. Entrega Ágil Disciplinada (Disciplined Agile Delivery) en notación de cadena de valor de Porter [7].

Debido a que el valor es creado solo cuando nuestros servicios están funcionando en producción, debemos asegurar que no solo entregamos en una vía rápida, sino que la implantación pueda ser realizada sin causar caos o interrupciones como pérdida de servicio, servicios incompatibles o fallas de seguridad o de cumplimiento de políticas [3].

DevOps extiende este acercamiento ágil a todo el ciclo de vida de entrega de software (Figura 9) y también a todos los interesados, incluyendo las líneas de negocio y los equipos de operaciones, en lugar de solo dejarlo para los equipos de desarrollo y QA [10].

El propósito original del movimiento DevOps fue remover las barreras entre los equipos de desarrollo y operaciones. La falta de comunicación y confianza entre estos equipos desafió su habilidad de desplegar software. La falta de comunicación efectiva resultante del trabajo en silos, hicieron difícil para los equipos de desarrollo y los interesados de negocio recibir la retroalimentación de los usuarios. Sin un mecanismo efectivo de retroalimentación de los usuarios, estos equipos están incapacitados para responder y beneficiarse de esta retroalimentación [10].

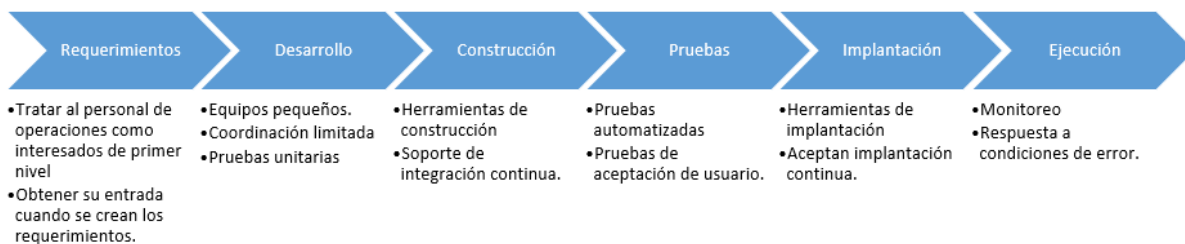


Figura 9. Cadena de valor de DevOps en notación de Porter [7].

Aunque el nombre “DevOps” sugiere que son capacidades orientadas a desarrollo y operaciones, este proceso es una capacidad empresarial que incluye a todos los interesados en una organización incluyendo los dueños del negocio, arquitectura, diseño, desarrollo, QA, operaciones, seguridad, partners y proveedores [5].

De esta manera llegamos a los principios de DevOps, los cuales son [5]:

- *Desarrolla y prueba contra ambientes similares a producción.* El objetivo es permitir a los equipos

de desarrollo y QA desarrollar y probar contra sistemas que se comportan de la misma manera que los sistemas de producción, así ellos pueden ver bien cómo la aplicación se comporta y rinde antes de estar lista para ser implantada.

- *Implanta con un proceso repetible y confiable.* La automatización es esencial para crear procesos que son iterativos, frecuentes, repetibles y confiables. La organización debe crear una cadena de entrega que permita la implantación continua y automatizada.
- *Monitorea y valida la calidad operacional.* El monitoreo frecuente provee alertas tempranas acerca de los errores operacionales y de calidad que pueden ocurrir en producción. Las métricas obtenidas de estos monitoreos deben estar en un formato en que todos los interesados del negocio las puedan entender y usar.
- *Amplifica los ciclos de retroalimentación.* Uno de los objetivos de DevOps es permitir a las organizaciones reaccionar y hacer cambios rápidamente. En la entrega de software, este objetivo requiere una organización que obtenga una retroalimentación rápida y luego aprenda rápidamente qué acción tomar. Este principio llama a la organización a crear canales de comunicación que permitan a los interesados acceder y actuar según retroalimentación:
 - Desarrollo debería ajustar sus planes de proyecto y prioridades.
 - Producción debería mejorar los ambientes de producción.
 - Negocio debería modificar sus planes de liberación de versiones.

2.3.2. Métodos de adopción de DevOps.

La Figura 10 presenta distintas formas por las cuales se puede adoptar este paradigma, a través de un modelo de una solución probada, usando un conjunto de métodos preferidos y capacidades [5].

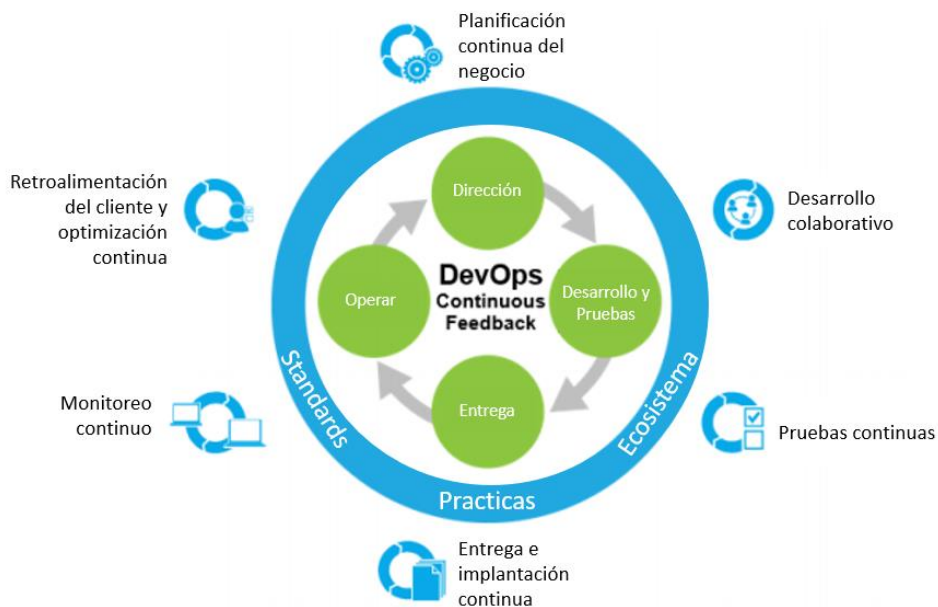


Figura 10. Arquitectura de referencia DevOps [5]

Dirección (Steer). Este camino de adopción es una práctica que se enfoca en establecer objetivos de negocio y ajustarlos basados en la retroalimentación del cliente: planificación continua de negocio [5].

Como se ha mencionado anteriormente, DevOps no es solo una práctica de TI. Las industrias IT de alto rendimiento son doblemente propensas a exceder las metas de ganancias, productividad y de participación en el mercado. Nicole Forsgren, Directora de Desempeño Organizacional y Análisis de CA Technologies, observa: “Puedes ver un cambio de la mirada de negocio sobre TI. Al principio se ve como un centro de costos, donde tienes que hacer software solo para mantener la operación, pero cuando empiezas a verlo como una oportunidad de diferenciarte que puede entregar valor genuino a los clientes, puedes atraer nuevos clientes y retener a los actuales” [11].

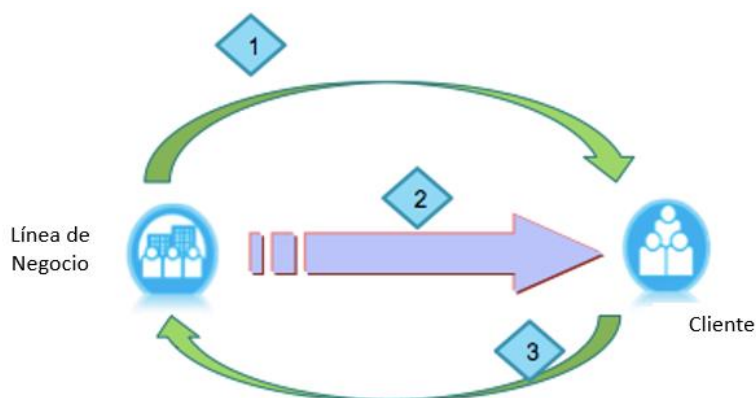


Figura 11. Releases rápidos para un retroalimentación de negocio rápido [10].

La Figura 11 muestra la interacción del área de negocio con el cliente. Primero se escucha las necesidades del cliente, luego se presenta un producto y finalmente se obtiene retroalimentación. El negocio necesita estar capacitado para reaccionar rápidamente a la retroalimentación del cliente. En consecuencia, muchas compañías usan técnicas de emprendimiento ágil. Estas técnicas involucran comenzar de a poco, identificando los resultados y los recursos necesarios para probar la visión o el valor del negocio y luego ir adaptándose y ajustándose continuamente basados en la retroalimentación del usuario [10].

Las entregas rápidas proveen al negocio una mayor agilidad, pero también se debe administrar con rapidez y confianza de que lo que se está entregando es lo correcto. No se puede entregar software rápidamente si no se confía en la exactitud de los objetivos de negocio, las métricas y las plataformas [5].

DevOps ayuda a reconciliar estas competencias, ayudando a los equipos a establecer objetivos de negocio a través de la cooperación y cambio continuo basado en el retroalimentación del cliente. Este acercamiento ayuda a los equipos a alcanzar un equilibrio a través de todas las fases del ciclo de vida de DevOps dentro del modelo de entrega continua [5].

Construcción y pruebas (Develop & Test). Este camino de adopción involucra dos prácticas: desarrollo colaborativo y pruebas continuas. Como se puede inferir, estos son el núcleo de desarrollo y QA [5].

Técnicamente, a través de integración continua podemos incluir estas dos prácticas, ya que al poseer un repositorio centralizado de versionado de fuentes y realizar pruebas automatizadas como se ha descrito anteriormente, es posible, además de la generación y liberación de versiones de manera constante, el trabajar de manera ágil (colaborativa) en equipos multidisciplinares.

Implantación (Deployment). Este camino de adopción es donde muchas de las capacidades claves de DevOps fueron originadas. Entrega continua (como se ha dicho anteriormente) toma el concepto de integración continua al siguiente nivel [5]. A través de entrega continua podemos construir software de tal manera que este puede ser liberado a producción en cualquier momento [4] de manera automática, reduciendo el riesgo al liberar nuevas versiones, asegurando repetitividad y por sobre todo, ayudando a obtener retroalimentación rápidamente [2]. La Figura 12 muestra de una manera idealista el proceso mencionado.

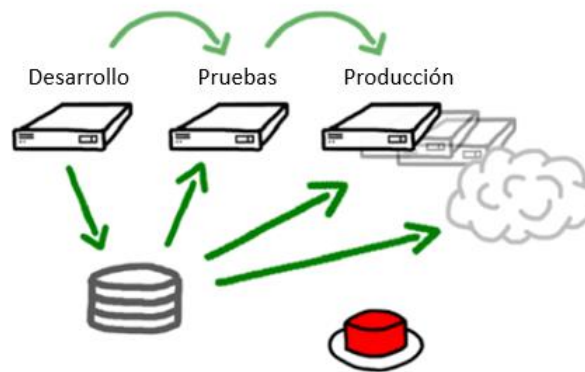


Figura 12 Representación idealista y simplista del proceso [2].

Operación (Operate). Este camino de adopción incluye dos prácticas que hacen posible al negocio monitorear cómo las entregas se comportan en producción y recibir retroalimentación de los clientes. Estos datos ayudan al negocio a reaccionar de manera ágil y adaptarlo a los planes de negocio y así obtener requerimientos y retroalimentación [10].

El monitoreo continuo provee datos y métricas a operaciones, QA, desarrollo, personal de las líneas de negocio y otros interesados acerca de las aplicaciones en diferentes estados del ciclo de entrega de software [5].

Estas métricas no solo están limitadas a producción. Algunas permiten a los interesados reaccionar mejorando o cambiando algunas características que serán entregadas y/o los planes de negocio requeridos para entregarlas [5].

Por ejemplo, para IT se podría habilitar retroalimentación en la cadena de valor de tecnología, usando un equivalente al cordón Andón de Toyota y sus respuestas relacionadas. El cordón Andón [12] se refiere a

una práctica creada en una planta de manufactura de Toyota: sobre cada centro de trabajo hay un cordón que todos los trabajadores y jefes pueden tirar cuando algo va mal, por ejemplo, cuando una parte está defectuosa, cuando una parte que se requiere no está disponible, o incluso cuando el trabajo toma más de lo que está documentado. La implementación de esta técnica también implica crear una cultura que lo haga seguro, e incluso con el valor de tirar el cordón Andón cuando algo salga mal, si esto ocurre cuando haya un incidente en producción o cuando el error ocurra antes en la cadena de valor, así como cuando alguien introduce un cambio que rompa la integración continua. Cuando las condiciones lleven a tirar este cordón, el equipo se reúne a resolver este problema y prevenir la introducción de nuevo trabajo hasta que la incidencia sea resuelta. Esto provee una retroalimentación rápida para todos en la cadena de valor (especialmente la persona que causó el problema), permite aislar rápidamente y diagnosticar el problema y prevenir factores futuros de complicación que tengan causas y efectos desconocidos [3].

Por otro lado, una retroalimentación importante que se puede obtener es cómo los usuarios navegan en la aplicación. Las nuevas tecnologías permiten al negocio obtener el comportamiento del cliente y sus puntos débiles a medida que utilizan la aplicación. Esta retroalimentación permite a los distintos interesados tomar acciones para mejorar sus aplicaciones y con ello la experiencia del cliente. Las líneas de negocio pueden ajustar sus planes de negocio, desarrollo puede ajustar su capacidad de entrega y operaciones puede mejorar los ambientes donde las aplicaciones están corriendo. Esta retroalimentación continua es un componente esencial de DevOps, permitiendo a negocio ser más ágil y responder a las necesidades del cliente [5]. La Figura 13 grafica hasta qué lugar de la cadena de desarrollo de software podríamos traer las preocupaciones de operaciones e incorporarlas tempranamente en el ciclo de desarrollo.

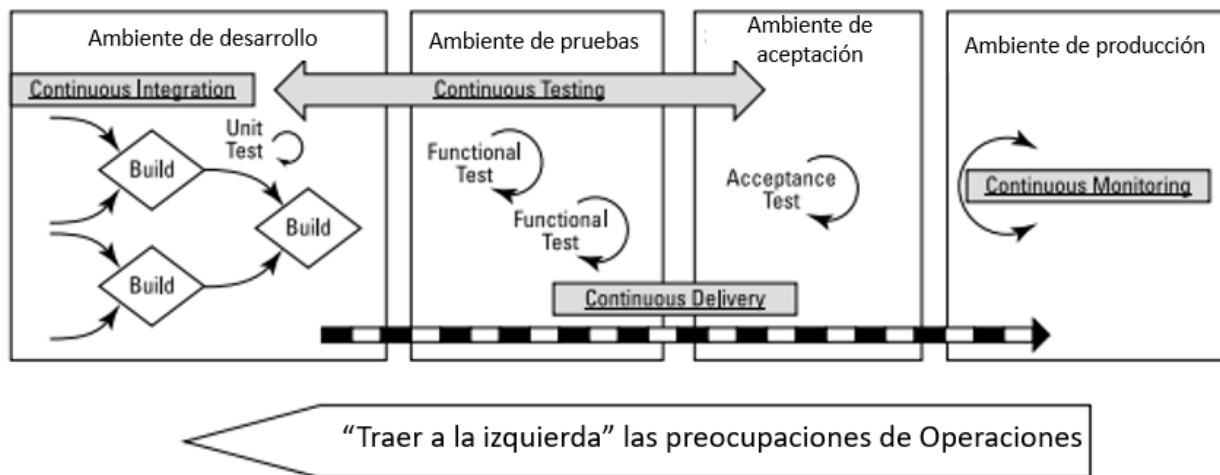


Figura 13. Mirada desde el punto de vista de operaciones [5].

2.4. Herramientas del ecosistema DevOps.

Para alcanzar el nivel de automatización descrito en la Sección 2.3, se necesitan distintos tipos de sistemas para: control de versiones, compilación y generación de ejecutables, ejecución de pruebas, implantación de las aplicaciones, ambientes para integración y otros sistemas para la interacción de estas aplicaciones en las distintas etapas y procesos que son parte de DevOps. Como se puede ver en la Figura 14, DevOps orquesta distintos sistemas operacionales y brinda información a las personas para detectar cambios, realizar funciones autónomas y notificar a los miembros del equipo del estado del release o el candidato a release y sus resultados [9].

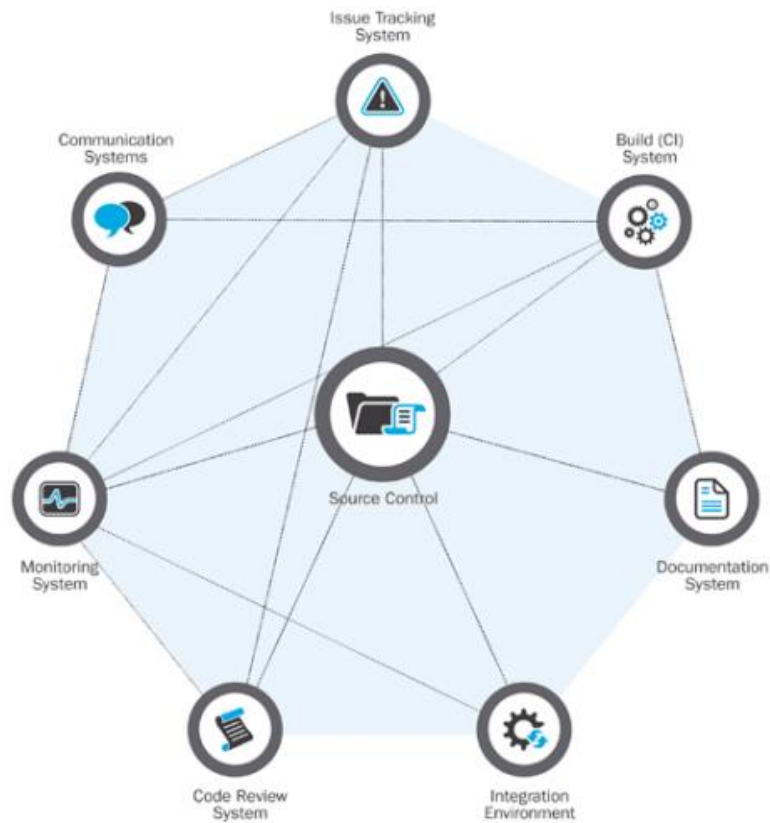


Figura 14. Aplicaciones del ecosistema DevOps.

3. Conceptualización de la solución

En este capítulo se describe la situación actual del proceso de desarrollo de aplicaciones del DCC, el estado actual de los desarrollos, cómo y quién desarrolla el software en el DCC, los procedimientos que siguen los involucrados en el desarrollo y los problemas que ha generado esta forma de trabajar.

Para finalizar, se realiza un levantamiento de los requisitos para crear una nueva arquitectura sobre la cual estará basado el modelo de desarrollo y operación del DCC que estará ajustada a la realidad del departamento.

3.1. Análisis del desarrollo en el DCC

Con el fin de conocer la situación del desarrollo de software en el DCC, se recopila información en dos dimensiones. Por un lado, se realiza un levantamiento de las aplicaciones con las que opera el DCC. Por otro, se captura el proceso de desarrollo de software en el DCC.

3.1.1. Aplicaciones tecnológicas

Para conocer el estado actual del desarrollo, se analiza la cantidad de aplicaciones desarrolladas y las tecnologías con las cuales se crean nuevas aplicaciones en el DCC.

Con respecto a la cantidad de aplicaciones a gobernar, la recién creada Área Aplicaciones realizó un levantamiento de aplicaciones, capturando de quién es la propiedad del desarrollo, si se encuentra en producción, y si apoya la operación del departamento, entre otros atributos. Se detectó un total de 34 aplicaciones. Existen 17 aplicaciones en las cuales el DCC es el propietario (esto es, que han sido construidas por el propio departamento) y que apoyan la operación. De estas 17 aplicaciones, 4 están en puesta en marcha, 2 están en prototipo y 11 se encuentran en producción. La Tabla 1 muestra a un extracto de los resultados, incluyendo solamente los atributos de interés para este trabajo de tesis y resaltando las 17 aplicaciones encontradas.

#	Nombre	Propiedad	Producción	Apoya a la operación del DCC
1	U-campus	Facultad	Producción	Sí
2	U-cursos	Facultad	Producción	Sí
3	PG Scopio	Externo	Producción	Sí
4	SAR	DCC	Producción	Sí
5	U-papers	DCC	Producción	Sí
6	U-fondos	DCC	Producción	Sí
7	Calificación académica	Universidad	Producción	Sí
8	Sitio web del DCC	DCC	Producción	Sí
9	U-proyectos	DCC	Producción	Sí

#	Nombre	Propiedad	Producción	Apoya a la operación del DCC
10	Intranet	DCC	Producción	Sí
11	U-productividades	DCC	Puesta en marcha	Sí
12	Administración de calendarios del PEC	DCC	Puesta en marcha	Sí
13	Mi UChile	Universidad	Producción	Sí
14	AUCAI	Universidad	Producción	Sí
15	Wiki DCC	DCC	Producción	Sí
16	Wiki Postgrado	DCC	Producción	Sí
17	U-dashboard	DCC	Prototipo	Sí
18	Inscripción MTI	DCC	Puesta en marcha	Sí
19	U-vinculación	DCC	Puesta en marcha	Sí
20	Bolsa de trabajo	CaDCC	Producción	Sí
21	U-fichas	DCC	Prototipo	Sí
22	Fondef	Externo	Producción	Sí
23	Coevaluaciones	DCC	Producción	Sí
24	Sistema de biblioteca	DCC	Producción	Sí
25	SAU	DCC	Producción	Sí
26	CASPLE	DCC	Producción	No
27	Discusión abierta	Universidad	Prototipo	No
28	Mainreq	DCC	Producción	No
29	Paguenpo	Externo	Producción	No
30	Preinscripción académica	DCC	Puesta en marcha	No
31	Redmine	DCC	Producción	No
32	Robótica y computación en Chile	DCC	Producción	No
33	Social Connector	DCC	Prototipo	No
34	Social Manager	DCC	Prototipo	No

Tabla 1. Estado de aplicaciones en el DCC.

Con respecto a la tecnología utilizada para el desarrollo de aplicaciones, se observa que el DCC no realiza desarrollos en una tecnología determinada o en un lenguaje estándar, sino que existe una gran variedad. En la Tabla 2, extraída del mismo levantamiento realizado por el Área Aplicaciones mencionado anteriormente, se pueden observar algunas de las tecnologías utilizadas. Para soportar esta gran variedad de tecnologías, el Área Sistemas crea máquinas virtuales según los requerimientos de cada una de las aplicaciones, tal como se lo indica cada equipo de desarrollo.

#	Sistema	Tecnología	BD	Modelo Arquitect.	Auten.	Autoriz.	Consume Inf.	API	Funcionalidades
4	Sistema de Administración de Recursos	PHP HTML Django 1.11	PostgreSQL	MVC	Propia	Propia	No	Sí	<ul style="list-style-type: none"> • Administración de recursos (salas, oficinas) del DCC.
5	U-papers	PHP (CodeIgniter)	MySQL	MVC	Propia	Propia	No	Sí	<ul style="list-style-type: none"> • Repositorio de publicaciones científicas de los académicos DCC • Permite realizar búsquedas por: título, fecha, tipo de publicación (artículo de journal, conferencia, libro) autor, palabras claves. • Estadísticas: Visitantes de la página, descargas de las publicaciones, publicaciones por autor, tiempo de permanencia en la página.
6	U-fondos	Joget Open SourceWork	MySQL	Cliente Servidor	Propia	Propia	n/d	No	<ul style="list-style-type: none"> • Gestión de asignación de fondos de investigación individual (FII) • Reporte de asignaciones asignadas por tiempo
8	Página Web DCC	Drupal	MySQL	n/d	No	No	No	No	<ul style="list-style-type: none"> • Mostrar información de diversa índole acerca de las actividades del DCC
9	U-proyectos	PHP 5.5 HTML 5 Javascript (jQuery, jQueryUI)	MySQL	MVC	SAU	SAU	n/d	Sí	<ul style="list-style-type: none"> • Registro de proyectos asociados a los académicos del departamento. • Información financiera y estadística de proyectos. • Registro de investigadores internos / externos que participaron en proyectos conjuntos con académicos del DCC. • Registro de académicos visitantes y postdoctorantes temporales en el DCC realizando estadías de trabajo.
10	Intranet	PHP	n/a	MVC	SAU	SAU	SAU	No	<ul style="list-style-type: none"> • Punto de acceso a varias aplicaciones del DCC. Provee un menú que permite navegar a aplicaciones específicas.
11	u-productividades	PHP (YII)	MySQL	MVC	SAU	SAU	n/d	No	<ul style="list-style-type: none"> • Ingreso, modificación y validación de productividades y coordinaciones del personal del DCC. • Informes de contabilidad y estadísticas del proceso de pago de productividades. • Administración de respaldo de datos de productividades. • Envío de información (mails) de productividades, validaciones pendientes.

#	Sistema	Tecnología	BD	Modelo Arquitect.	Auten.	Autoriz.	Consume Inf.	API	Funcionalidades
12	Admin. Calendarios PEC	Node.js 4 HTML 5 jQuery 2.2.3 Express 4.13.4 Bootstrap 3 JavaScript	MongoDB	n/d	Propia	Propia	n/d	No	<ul style="list-style-type: none"> • Facilitar la calendarización de clases de los diferentes programas de postgrado dentro del DCC • Administración de salas, cursos, docentes, disponibilidad de docentes, programas, calendarios por programa • Exportar calendarios en formato PDF
15	Wiki DCC	Dokuwiki	n/a	n/d	Propia	Propia	No	No	<ul style="list-style-type: none"> • Recolectar y mostrar información general del departamento
16	Wiki Postgrado	Dokuwiki	n/a	n/d	Propia	Propia	No	No	<ul style="list-style-type: none"> • Recolectar y mostrar información sobre los programas de postgrado, principalmente el Doctorado
17	U-dashboard	PHP 5.6 CodeIgniter 3.0 Html 5 Javascript 1.8	MariaDB	MVC	Propia	U-passaporte	n/d	No	<ul style="list-style-type: none"> • Almacenamiento de métricas de negocio y/o productividad para las áreas y unidades del DCC. • Agregación, modificación y visualización de métricas • Asignación de permisos en diferentes roles para la edición de la información.
18	Inscripción MTI	Java Tomcat	n/d	MVC	Propia	Propia	No	No	<ul style="list-style-type: none"> • Realización y seguimiento de postulaciones al MTI
19	U-Vinculación	Python (Django)	MySQL	MVC	Propia	Propia	U-Campus	No	<ul style="list-style-type: none"> • Captura y visualiza información sobre egresados del departamento
21	U-fichas	Python (Django)	PostgreSQL SQLite	n/d	Propia	Propia	PGScopio U-papers U-proyec.	No	<ul style="list-style-type: none"> • Generar fichas académicas para cada uno de los docentes integrantes de los programas de postgrado para su acreditación • Cada ficha incluye: información personal, tesis dirigidas (DCC y externas), publicaciones realizadas, proyectos y asesorías en los que haya participado para distintos periodos. • Extracción de datos de fuentes de verdad como: PGscopio (MTI, MSC, Doctorado), U-papers, U-proyectos.
23	Coevaluaciones	Python (django)	PostgreSQL	MVC	Propia	Propia	No	No	<ul style="list-style-type: none"> • Realización de co-evaluaciones entre estudiantes para distintos cursos del pregrado.
24	Sistema de biblioteca	Racket	n/a	n/d	n/d	n/d	No	No	<ul style="list-style-type: none"> • Solicitudes de libros para la biblioteca del departamento.
25	SAU	PHP	MySQL	n/d	SAU	SAU	n/d	Sí	<ul style="list-style-type: none"> • Manejo de credenciales de acceso (autenticación) y permisos (autorización) de acuerdo a roles para ciertas aplicaciones.

Tabla 2. Aplicaciones y tecnologías.

El análisis de las principales funciones que proveen estas aplicaciones permite detectar que éstas no son suficientes para dar apoyo a todas las actividades o el ámbito de operación completo del DCC, por lo que, a futuro, será necesario no solo mantener estas aplicaciones ya desarrolladas, sino que también construir o adquirir más aplicaciones.

3.1.2. Proceso de desarrollo

A continuación, se explica el proceso de desarrollo actual, a alto nivel, indicando quiénes participan en él y el tiempo dedicado al desarrollo, cuál es el ciclo de vida de las aplicaciones en el DCC y cuáles son los artefactos de entrada y salida.

Existen distintos actores que participan en el proceso de desarrollo de software en el DCC. Es posible observar que la fuerza de desarrollo proviene principalmente de estudiantes que por lo general destinan un tiempo acotado, delimitado por su permanencia en la carrera en la cual están realizando el desarrollo, tales como alumnos del curso Ingeniería de Software II, memoristas y tesistas de MTI. También existen casos en los cuales académicos del departamento han dedicado parte de su tiempo al desarrollo de aplicaciones.

Respecto al ciclo de vida de las aplicaciones, si bien no existe un procedimiento formal de desarrollo de software (a excepción del curso Ingeniería de Software II que sí tiene un proceso definido) existen ciertos procedimientos que se activan una vez cumplidos algunos hitos. Por ejemplo, existe el concepto de entregable (aplicación compilada y en condiciones de ser instalada en producción) y de entrega de requerimientos, pero no hay una etapa formal de pruebas ni tampoco de instalación.

Los artefactos del proceso actual corresponden principalmente a las tesis entregadas por los alumnos memoristas y tesistas, y por parte del curso Ingeniería de software II, la documentación registrada en la aplicación MainReq que es la que utilizan los profesores del curso para monitorear el avance de los proyectos. A su vez, también está el ejecutable en caso de alcanzar este hito. Sin embargo, no existe un repositorio de fuentes ni tampoco un repositorio de información centralizado del departamento.

3.2. Entrevistas a los interesados

Para corroborar las observaciones realizadas y capturar en mayor detalle la realidad del departamento, se realizan entrevistas a quienes actualmente participan en el proceso de desarrollo, puesta en producción y mantenimiento. Las personas entrevistadas son: académico responsable del curso Ingeniería de Software II, jefe de Área Sistemas y a un miembro del Área Sistemas que está a cargo del soporte de primera línea de las aplicaciones.

3.2.1. Protocolo

Se realiza una entrevista semiestructurada [13], en la que el entrevistador posee una guía con preguntas o puntos clave a cubrir por la entrevista, pero tiene la libertad de hacer preguntas complementarias o

cambiar el orden de las preguntas dependiendo de cómo evoluciona la entrevista. Gran parte de la información que se obtiene de las entrevistas semiestructuradas es cualitativa, por lo que no se suele utilizar un conjunto de respuestas predefinidas. Las entrevistas de este tipo requieren que el entrevistador posea mayor conocimiento del tema que en una entrevista estructurada. Debido a que parte de las preguntas son abiertas y la entrevista puede divergir de la guía, es recomendable esta sea grabada y luego transcrita.

La Tabla 3 detalla el protocolo definido para las entrevistas.

Preámbulo	Se le consulta al entrevistado si es posible grabar la conversación y se le comenta que la información obtenida en esta entrevista será utilizada en el contexto de la tesis de magíster del entrevistador y del Área Aplicaciones del DCC.
General	<p>Objetivo</p> <p>Obtener información acerca de cómo se está llevando actualmente el proceso interno de desarrollo de software, desde el momento de la toma de requerimientos hasta el posterior soporte y mantención de la aplicación.</p>
Procedimiento	<p>Características de los entrevistados</p> <p>Por parte de desarrollo, se entrevista al académico responsable del curso Ingeniería de Software II, en el cual se construyen algunos de los sistemas del DCC. Cabe señalar que este académico es el que ha dirigido la mayor parte de las memorias de pregrado y MTI en las cuales han desarrollado software para el DCC.</p> <p>Por parte de operaciones, se entrevista al encargado del Área Sistemas, área que es responsable de proveer la infraestructura necesaria para la correcta ejecución de las aplicaciones entregadas por los estudiantes. Además, se entrevista a un miembro del área encargado del soporte de primera línea.</p> <p>Cantidad de entrevistados</p> <p>Se entrevista a 3 personas.</p> <p>Duración de la entrevista</p> <p>Entre 15 y 30 minutos.</p> <p>Equipo y material</p> <p>Se utiliza un teléfono celular para grabar las entrevistas. Se provee al entrevistado la pauta para su análisis y se anotan en un cuaderno aspectos que podrían resultar relevantes como complemento a las grabaciones.</p>
Instrumento	<p>Guía de la entrevista</p> <p>Entrevista al académico a cargo del desarrollo de aplicaciones.</p> <ul style="list-style-type: none"> • ¿Quién o quiénes son los encargados de solicitar cambios o nuevas funcionalidades?

	<ul style="list-style-type: none"> • Al momento de recibir un nuevo requerimiento, ¿éste queda guardado en alguna plataforma digital? ¿De qué manera se puede visualizar el trabajo realizado en lo referente a diagnóstico, problemas encontrados, etc.? • Durante la fase de pruebas, ¿de dónde se obtienen los datos de prueba o cuáles son las bases de datos y servidores que se utilizan para realizar pruebas? • Al momento de terminar el desarrollo de una aplicación: <ul style="list-style-type: none"> ○ ¿Cuáles son los entregables? (ejecutable, documentación, etc.) ○ ¿Quién o quiénes gestionan el código fuente del proyecto? ○ ¿Cómo es el proceso de instalación en un ambiente productivo? • ¿Conocen o están familiarizados con el procedimiento de soporte en caso de incidencias sobre el código?, ¿Cómo y quiénes registran las incidencias? <p>Entrevista al Área Sistemas:</p> <ul style="list-style-type: none"> • Al momento de recibir por parte de desarrollo una nueva aplicación (o evolución de una aplicación), ¿cómo es el proceso o procedimiento de instalación? • Una vez instalado el sistema en producción, ¿cuál es el proceso o procedimiento de soporte sobre esa aplicación? ¿Existe una mesa de ayuda o de soporte? • En el caso de mantenciones correctivas a las aplicaciones, ¿quién realiza el diagnóstico y las modificaciones necesarias en las aplicaciones?
--	--

Tabla 3. Entrevista a Interesados en el proceso de desarrollo.

3.2.2. Ejecución

Las entrevistas fueron realizadas en el mes de junio de 2018. Se realizaron cada una por separado, al encargado del Área Sistemas, al encargado de soporte de primera línea y al encargado de coordinación de desarrollo de aplicaciones del DCC. Se utilizó la pauta como instrumento guía, pero se permite al entrevistado ahondar más en los temas que considere importantes al momento de la entrevista.

3.2.3. Resultados

Las entrevistas a los miembros del Área Sistemas arrojaron resultados coincidentes. A continuación, se listan las respuestas más relevantes, agrupando las de ambos entrevistados.

- 1) Ambos entrevistados coinciden en que no existe un procedimiento de entrega definido, declarando que al momento de ocurrir un nuevo desarrollo se entrega por parte del Área Sistemas una máquina virtual (“virtual host”) la cual contendrá las configuraciones necesarias para que los desarrolladores (usualmente estudiantes) puedan realizar su trabajo. No existe restricción de acceso a esta máquina, es decir, se entregan credenciales con acceso completo.

Cabe señalar también que el procedimiento de limpieza (dar de baja) de las máquinas en desuso

corresponde a una tarea realizada por el Área Sistemas y no a una solicitud de desarrollo. Esto tiene como consecuencia que el Área Sistemas podría dar de baja una máquina virtual de una aplicación que está en uso, o mantener disponible una aplicación por mucho más tiempo que su vida útil.

- 2) Ambos entrevistados coinciden en que es el Área Sistemas quién realiza el soporte de primera línea a través de una persona encargada de ello. Este trabajo incluye desde la modificación de datos hasta la modificación de código (soporte correctivo). En este último punto, se consideran mayormente los lenguajes que no están dentro de un ejecutable debido a la facilidad y a la disponibilidad del código.

Si dentro del análisis realizado al código se encuentra alguna funcionalidad que no tenga explicación dentro del contexto de la aplicación y es ésta la que está provocando fallos, entonces se avisa al académico responsable del desarrollo de aplicaciones.

- 3) En el caso de las mejoras, ambos entrevistados coinciden en que los desarrollos que requieran cambiar funcionalidades de software son derivados al académico responsable del desarrollo de aplicaciones. Sin embargo, si luego de un análisis se determina que este cambio en la funcionalidad no requiere de muchos cambios a nivel de código, éste también puede ser tomado por el encargado de soporte de primera línea del Área Sistemas.

La entrevista al encargado del desarrollo arrojó los siguientes resultados:

- 1) El entrevistado afirma que son los mismos usuarios, o bien el Área Sistemas a través de alguna detección de incidencia, los que realizan las solicitudes las cuales son “archivadas” (en un correo electrónico) y luego derivadas a los distintos equipos de desarrollo. Además, el entrevistado afirma que no existe un equipo de desarrollo fijo, por lo que las solicitudes son derivadas según disponibilidad.
- 2) Al ser consultado por la persistencia de los requerimientos, la respuesta es que estos no quedan en ninguna plataforma, aunque en el contexto del curso Ingeniería de Software II sí existe una herramienta formal llamada MainReq. En esta herramienta se almacenan los requerimientos dentro del contexto del trabajo que realizará el grupo de estudiantes en el curso, pero no es un repositorio o backlog del total de solicitudes del DCC. Además, el entrevistado indica que tampoco existe una herramienta de software que se encargue de la gestión dentro de la cadena de entrega del software.
- 3) Con respecto a los datos y ambientes de prueba, el entrevistado comenta que estos son proporcionados por el Área Sistemas, reafirmando que son ellos quienes configuran y entregan las cuentas a los estudiantes para que puedan acceder. Estos ambientes vienen con las bases de datos y los datos pre-cargados cuando es un mantenimiento. En el contexto del curso se incluye una etapa de pruebas, las cuales son funcionales (se prueban las principales funciones de la aplicación), de inspección de código (se revisa el uso de buenas prácticas), y además de pruebas de aceptación de usuario (principalmente realizadas por el profesor del curso y no por los usuarios).

finales).

- 4) Al momento de terminar el desarrollo de software, el entrevistado comenta que:
 - a. Se entrega el código fuente (en Git o versionado de alguna manera), se entrega el ejecutable y la documentación en MainReq.
 - b. No existen personas o áreas dedicadas a la gestión del código fuente.
 - c. El Área Sistemas es quien realiza los procesos de instalación de las aplicaciones.
- 5) El entrevistado dice que no existe un procedimiento de soporte de aplicaciones.

3.2.4. Análisis de la situación actual

Para el Área Sistemas, si bien ambos entrevistados declaran que no existe un procedimiento formal, este pareciera estar implícito ya que también declaran que, a solicitud del curso o desarrollador, es la propia área la encargada de generar una máquina virtual para pruebas con las de bases de datos correspondientes. Se observa también que, al dar acceso completo a la máquina, podría haber problemas de seguridad en cuanto a la libertad que se da al estudiante de controlar un espacio alojado dentro del DCC. Sin perjuicio de ser una “caja de arena”, esta podría eventualmente ser utilizada para fines que no sean conducentes a los objetivos del curso o contexto en el cual se concedió el acceso.

Por otro lado, se observa que no hay una gestión de ambientes o máquinas virtuales en uso/desuso ya que, según las afirmaciones de los entrevistados, es al momento de realizar la limpieza (o dada de baja) de los mismos se reciben reclamos si esto causa que alguna aplicación no esté disponible.

De esto último también se puede deducir que para algunas aplicaciones no hay una separación entre los ambientes de prueba (objetivo con el cual el Área Sistemas declara que se proveen estos ambientes a los desarrolladores) y de producción.

Con respecto a la de gestión de requerimientos por parte de desarrollo, se puede observar que si bien existe gestión de requerimientos para el curso de Ingeniería de Software II, para los demás equipos que realizan desarrollo (memoristas, tesis de MTI, entre otros) esto no es así, quedando estos datos en los informes finales de cada uno de los desarrolladores. Además, según los datos aportados por el mismo entrevistado, esta información no es actualizada y queda solo en su versión final en el contexto del curso, por lo que cada corrección o mejora no es documentada y por lo tanto la información de MainReq puede no necesariamente ser consistente con lo que está realmente desarrollado.

Además, se puede observar que no existe un repositorio centralizado de versionado de código fuente, sino que los entregables pueden venir en un repositorio de libre elección por parte del grupo desarrollador ya que, según se declara, no hay personas o áreas dedicadas a la gestión del código fuente.

3.3. Alcance y expectativas

En esta sección se describe el alcance y las expectativas sobre el proceso de gestión y desarrollo y la arquitectura de la plataforma tecnológica que lo asiste, en función de los hallazgos obtenidos en el relevamiento y en las entrevistas, y de la capacidad de recursos del DCC. Para documentar el alcance, se utiliza la técnica descrita en el Capítulo 8 del libro “*Software Systems Architecture*” [14]. Primero se identifican los objetivos de negocio a cubrir con la implementación de este proceso. Luego se define el alcance y las áreas principales que se abarcarán, indicando los sistemas que se darán de baja con esta nueva implementación. Finalmente se identifican a los distintos interesados y sus áreas de interés y cómo estos actores interactúan con el nuevo proceso.

3.3.1. Objetivos de negocio y motivadores

Los objetivos de negocio y motivadores dan el contexto para el proyecto y son la razón fundamental para que el proyecto exista. Un objetivo de negocio es un objetivo específico que la organización tiene, mientras que un motivador de negocio es alguna fuerza que actúa en la organización que requiere que se comporte de alguna forma en particular con el fin de proteger y hacer crecer el negocio.

La importancia de entender los objetivos y motivadores radica en que el proceso y plataforma a desarrollar debe asistir a la organización para alcanzar estos desafíos

En el caso particular de la construcción de aplicaciones en el DCC, se definen los siguientes objetivos de negocio y motivadores:

- *Falta de gobernanza sobre las aplicaciones*: al ser una entidad experta en el desarrollo de aplicaciones, tanto en la parte técnica como en la gestión de los mismos, se desea aplicar estos conocimientos sobre los desarrollos internos.
- *Falta de procedimientos (desarrollo de software)*: se quiere aclarar en mayor detalle las etapas de desarrollo y operación de las aplicaciones, con el fin de documentar los artefactos entregables de cada una de ellas y a través de esto, generar repetibilidad en este proceso.
- *Falta de procedimientos (soporte de las aplicaciones)*: se requiere crear un mecanismo de soporte para que los usuarios de las aplicaciones puedan reportar los errores ocurridos durante su uso, así como también ir un paso más allá y adelantarse a estos reportes a través de un monitoreo proactivo.

3.3.2. Alcance

La descripción del alcance del proceso y de la plataforma de desarrollo están organizados en términos de las principales áreas funcionales que debe soportar, las interfaces y sistemas externos con los que debe interoperar, los sistemas y datos legados que se darán de baja cuando comience a operar, las restricciones en la construcción y operación tanto del proceso como de la plataforma y los principios que debe respetar para alcanzar el estado futuro deseado.

Áreas funcionales

A alto nivel, el proceso de desarrollo de software del DCC debe considerar las siguientes áreas funcionales: recolección, desarrollo, aceptación, instalación y monitoreo. Estas áreas funcionales están basadas en la literatura sobre DevOps, tal como se describe en la Sección 2.3.1, y conforman las principales etapas del pipeline de entrega de aplicaciones del DCC definido en el Capítulo 4.

El área funcional de recolección abarca el registro, gestión y planificación de las ideas e incidencias reportadas por los usuarios, desarrolladores, infraestructura (sistemas) y cualquier interesado en el desarrollo de aplicaciones del DCC. En detalle, cubre lo siguiente:

- F.1. La plataforma provee un mecanismo de gestión de tickets (incidencias).
- F.2. La plataforma provee un mecanismo de registro y gestión de solicitud de nuevas funcionalidades.
- F.3. La plataforma provee un mecanismo para el registro y monitoreo del ciclo de vida de desarrollo y operación de las aplicaciones.

El área funcional de desarrollo corresponde a la de creación de código fuente y pruebas unitarias, e involucra las siguientes funcionalidades:

- D.1. La plataforma provee un mecanismo de versionado de código fuente.
- D.2. La plataforma provee un ambiente de desarrollo.
- D.3. La plataforma provee un repositorio de bibliotecas disponibles para los desarrolladores.
- D.4. La plataforma provee un servidor de integración de forma de asegurar que el código en el sistema de versionado pueda ser compilado para construir un ejecutable.
- D.5. La plataforma provee un ambiente para que los desarrolladores puedan administrar y que los usuarios puedan usar para realizar pruebas antes de pasar a los ambientes de aceptación.

El área funcional de aceptación corresponde a la integración con otros sistemas y pruebas de usuario de las aplicaciones desarrolladas. El detalle de las funcionalidades es:

- A.1. La plataforma provee un ambiente para verificar la correcta integración con otros sistemas.
- A.2. La plataforma provee un ambiente para pruebas de aceptación de usuario, de manera tal que sea posible verificar que la aplicación funcione según lo esperado.

El área funcional de instalación incluye a las etapas de entrega y despliegue del ejecutable, esto se define como:

- I.1. La plataforma provee la capacidad de realizar instalaciones a pedido o bien automáticas en los diferentes ambientes.

El área funcional de monitoreo incluye las etapas en las cuales las aplicaciones se encuentran disponibles

en el ambiente de producción.

- M.1. La plataforma dispara alertas al detectar errores en tiempo de ejecución de las aplicaciones instaladas.
- M.2. La plataforma provee un mecanismo que permite el registro de incidencias durante su uso.

Interfaces y sistemas externos

La plataforma tecnológica que dé soporte al proceso de gestión y desarrollo del departamento estará compuesta por varias herramientas (desde herramientas de captura de ideas e incidencias hasta la puesta en producción de los sistemas). Los desarrolladores, al seguir el proceso de desarrollo, interactuarán con estas herramientas usando las interfaces que éstas dispongan. Específicamente, las interfaces más relevantes son las siguientes.

- *S.1. Comunicación del nuevo requerimiento a través de la interfaz del sistema de tickets.* En este punto la interfaz es utilizada por los interesados, por el dueño del producto o sistema a modificar y por los desarrolladores.
- *S.2. Comunicación con los desarrolladores a través del sistema de versionado de código fuente.* Para trabajar con el código fuente, se debe seguir el protocolo del sistema de versionado, utilizando una aplicación cliente y siguiendo un flujo de trabajo específico.
- *S.3. Comunicación con la herramienta de automatización de la implantación de las aplicaciones.* A través de la interfaz del sistema operativo, se ejecuta la herramienta de automatización de implantación de aplicaciones la cual recibe estos comandos e instala la aplicación indicada en el ambiente señalado.
- *S.4. Comunicación con los administradores de plataforma y desarrolladores (si aplica) al momento de generar alertas en caso de fallas del software.* A través del sistema de monitoreo se levantarán alertas y éstas serán enviadas tanto a administradores como al soporte correspondiente (desarrolladores, administradores de plataforma).

No se detectan otras interfaces o sistemas de sistemas que están actualmente en operación en el DCC, debido principalmente a que la plataforma es nueva en su alcance e implementación.

Sistemas y datos legados

Los sistemas y datos legados son aquellos que serán dados de baja debido a la implementación del proceso y plataforma. A pesar de no ser una preocupación directa, es importante tenerlo en consideración ya que reemplazar un sistema en algunos casos puede ser más difícil incluso que crear uno desde cero.

En el DCC existe en la actualidad un servidor SVN de versionado de código fuente el cual será sustituido por una nueva herramienta del mismo tipo.

En cuanto a los datos que serán migrados a la nueva plataforma, se detectan los siguientes:

- Código fuente de las aplicaciones actuales.
- Documentación existente de los sistemas.
- Incidencias existentes.
- Requerimientos y necesidades funcionales.

Limitaciones

El proceso de gestión y desarrollo y la plataforma tecnológica que le dé soporte no contempla todas las técnicas propuestas en DevOps, sino que incluye solo aquellas que tienen mayor posibilidad de ser adoptadas por el DCC en el corto plazo. Las siguientes técnicas están fuera del alcance del proceso definido, y se propone como trabajo futuro considerar su inclusión:

- Pruebas automatizadas, incluyendo pruebas unitarias, pruebas de “humo”, pruebas de integración y flujos funcionales automatizados.
- Plataforma de comunicación entre los desarrolladores.
- Plataforma de comunicación entre el equipo de desarrollo y los usuarios.
- Uso de un orquestador para integración, entrega e implantación continua.

Restricciones

Las restricciones son aspectos invariantes en el funcionamiento de la organización dentro de los cuales se debe enmarcar el desarrollo y operación de la nueva arquitectura. En el caso del DCC, se detectaron las siguientes restricciones:

- *R.1. Los equipos de desarrollo deben seguir siendo los mismos.* El proceso debe adecuarse a los equipos de trabajo existentes en la actualidad, en cuanto a experiencia y tiempo de dedicación al desarrollo de aplicaciones.
- *R.2. La metodología de desarrollo no puede estar restringida.* Si bien es cierto que para el desarrollo de aplicaciones utilizando el paradigma DevOps se recomienda utilizar metodologías ágiles de desarrollo, se debe crear un proceso que permita que los cursos y/o personas que la utilicen sean libres de escoger la metodología de desarrollo que es más apropiada a su contexto.
- *R.3. El equipo del Área Sistemas debe poder usar la nueva plataforma.* Si bien la interacción del Área Sistemas será a través de ciertas herramientas, éstas deben ser adoptables por el área.

Principios

Los principios son aquellas frases fundamentales de confianza o enfoque que guía la definición arquitectónica. Puede estar basado en el contexto actual o en un estado futuro deseado. Para el caso del DCC, el proceso estará basado en los siguientes estados futuros deseados:

- *P.1. Centralización de la información.* En la actualidad, no se tiene la información en un solo repositorio común, ya sean las incidencias reportadas, el código fuente, la documentación, etc.
- *P.2. Proceso soportado por herramientas colaborativas.* Al estar los desarrollos dentro de un contexto de tiempo limitado, alcanzando este principio se podrán continuar o actualizar desarrollos independientes del tiempo y personas en el proyecto.
- *P.3. Potenciar la automatización de tareas.* Al automatizar todo lo que sea posible, se reducirá el riesgo conducente a error de las tareas repetitivas.

3.3.3. Interesados

Un interesado (stakeholder en inglés) es una persona, grupo o entidad que tiene interés o preocupación acerca de la realización de la arquitectura o del software.

Para saber que se ha detectado un interesado válido, este debe contar con las siguientes características:

- *Debe estar informado:* debe tener la experiencia y el entendimiento necesario para tomar las decisiones correctas
- *Debe estar comprometido:* deben estar disponibles para participar en el proceso y preparados para tomar decisiones posiblemente difíciles.
- *Deben tener autoridad:* hay que estar seguro de que las decisiones tomadas por los stakeholders no serán revocadas más adelante (a un costo potencialmente alto).
- *Debe ser representativo:* si un stakeholder es un grupo, seleccionar a alguien que no represente sólo sus intereses, sino que represente al grupo.

A su vez, también es posible clasificar a los interesados según sus roles y preocupaciones dentro del contexto del proyecto. Estos incluyen a interesados no pertenecientes a las áreas de tecnología (como adquirientes y usuarios) y los pertenecientes a áreas de tecnología (como desarrolladores, administradores de sistemas y soporte). Podemos encontrar entonces adquirientes, asesores, comunicadores, desarrolladores, mantenedores, proveedores, administradores de sistema, testers y usuarios [14].

Las siguientes entidades y grupos de usuarios cumplen con los atributos definidos para ser definidos como interesados válidos del proceso y la plataforma tecnológica:

- *Área Sistemas:* la componen los integrantes del Área Sistemas, quienes son los encargados de administrar las plataformas y mantener disponibles las aplicaciones. Debido a estas responsabilidades, el Área Sistemas cumple con las características de ser mantenedor, usuario, administradores de sistemas y proveedores del servicio.
- *Área Aplicaciones:* la componen distintos actores quienes gestionan los desarrollos solicitados. Son los encargados de la gestión de nuevos desarrollos y de la evolución de los desarrollos actuales. La interacción del Área Aplicaciones con la plataforma los caracteriza

- como desarrolladores, mantenedores, administradores de sistema y comunicadores.
- *Equipos de desarrollo*: lo componen alumnos, académicos y otros quienes crean código a partir de las necesidades expuestas por los usuarios al Área Aplicaciones. Los desarrolladores tienen las características de usuario ya que interactúan con la plataforma a través de sus interfaces.
 - *Usuarios*: lo componen académicos, administrativos y otros quienes generan necesidades. Al igual que los equipos de desarrollo, este grupo tiene las características de usuario exponiendo sus necesidades a través de una de las interfaces de la plataforma.
 - *Dirección del DCC*: lo componen académicos quienes están interesados en la gobernanza de TI. Son adquirientes y usuarios.

4. Proceso y Plataforma de Desarrollo

En este capítulo se describe el proceso de desarrollo y gestión definido para el DCC, junto con la plataforma tecnológica que asiste a este proceso.

El proceso define un pipeline de desarrollo, entrega, implantación y monitoreo de aplicaciones, usando técnicas del paradigma de DevOps descritas en el Capítulo 2. Sin embargo, no todas las prácticas de este paradigma han sido incorporadas. El relevamiento documentado en el Capítulo 3 muestra que hay una necesidad de gestión y gobernanza que es aún más relevante que aplicar técnicas ingeniería continua (esto es integración continua, entrega continua, implantación continua, monitoreo continuo). Debido a la necesidad de preservar el mecanismo de conformación de equipos y permitir que éstos organicen su forma de trabajo de acuerdo al contexto en el que se encuentran (ver las restricciones en la Sección 3.3.2), el foco del proceso está en la definición de un pipeline que provea esta flexibilidad, pero que al mismo tiempo, imponga algunas prácticas que le permita a las Áreas Aplicaciones y Sistemas operar más coordinadamente y lograr las expectativas definidas en la Sección 3.3.

Asistir el proceso con herramientas y favorecer la automatización son prácticas recomendables por el paradigma DevOps, siendo éste incluso uno de sus caminos de adopción (Sección 2.3.2). Como se comentó anteriormente, en el caso del DCC se observa que la falta de gobernanza es una de las principales necesidades, por lo tanto, éste debe ser el foco de la plataforma tecnológica que asista el proceso.

La Sección 4.1 describe el proceso de desarrollo definido para el DCC. La Sección 4.2 documenta la plataforma tecnológica que da soporte al proceso definido. La Sección 4.3 cierra el capítulo con un análisis de la cobertura del proceso y la plataforma con respecto a los objetivos y motivadores definidos en la Sección 3.3.1 y las áreas funcionales definidas en la Sección 3.3.2.

4.1. Proceso de desarrollo

En esta sección se describe el proceso de desarrollo del DCC. Primero se define las etapas que conforman el pipeline de desarrollo y operación de aplicaciones del DCC. Luego, en base al pipeline, se definen los principales roles que participan del proceso, así como los artefactos producidos y utilizados en las etapas del pipeline.

4.1.1. Pipeline del DCC

Para organizar el ciclo de vida de las aplicaciones en el DCC, se define un pipeline a través del cual se convierten las necesidades en piezas de código, que a su vez formarán parte de aplicaciones que se dejan disponibles para los usuarios. En la Figura 15 se muestra el orden de las etapas que componen el pipeline definido.



Figura 15. Pipeline propuesto para el DCC.

Etapas y tareas del pipeline

Cada etapa del pipeline corresponde a una o más tareas que ejecutan los actores, ya sea para obtener o generar artefactos que se utilizarán en las etapas siguientes. A continuación, se realiza una descripción de las tareas que se realizan en cada etapa del pipeline.

Recolección. Esta es la etapa de entrada al pipeline. En ella se capturan las necesidades, ideas e incidencias, tanto de los usuarios como de desarrolladores u otros interesados. El resultado de esta actividad es la creación de un backlog de potenciales requisitos a desarrollar.

Plan. Esta etapa corresponde a la selección, descarte, agrupamiento y priorización de los requisitos del backlog, que dan lugar al inicio de nuevos proyectos de desarrollo. Además, en este punto se solicita la creación de los ambientes necesarios para la ejecución del nuevo proyecto; estos son un ambiente de desarrollo y un ambiente de pruebas funcionales. Una vez realizada esta tarea, se refinan los requerimientos que serán abordados en el proyecto.

Desarrollo. En esta etapa se analizan e interpretan los requerimientos, y se implementa su resolución mediante la creación o actualización del código fuente generando así nuevas versiones de las aplicaciones y componentes involucradas. Estas nuevas versiones son consideradas *snapshots*, esto es, versiones que deben ser probadas y que aún no están en calidad de producción. El desarrollo se realiza en las máquinas locales de los desarrolladores. Sin embargo, al publicar las piezas de código en un controlador de versiones, el desarrollador genera un *snapshot* que puede ser instalado en ambiente de desarrollo con el fin de realizar pruebas funcionales antes de liberar a ambiente de pruebas.

Pruebas. Esta etapa es paralela a la etapa de desarrollo. Se realizan las primeras pruebas funcionales por parte del usuario, o bien del mismo equipo de desarrollo. Para ello, los desarrolladores instalan el/los *snapshots* de los componentes desarrollados en el ambiente de pruebas y lo dejan disponible a los interesados. De esta etapa se podrían obtener incidencias o nuevos requerimientos para el proyecto.

Integración. Esta etapa tiene como objetivo confirmar la correcta integración de la aplicación en desarrollo con los componentes con las cuales tenga o genere dependencias. Para ello se instala el/los *snapshots* ya probados en el ambiente de integración. Acá, al igual que en la etapa anterior, eventualmente se podrían levantar incidencias, tanto funcionales como técnicas.

Aceptación. Esta etapa consiste en la realización de pruebas funcionales a la aplicación candidata a release luego de haber completado sus pruebas de integración. Para la ejecución de estas pruebas, se instala la aplicación en el ambiente de aceptación dejándola disponible para los usuarios. Al igual que en las etapas anteriores, de esta etapa también podrían generarse incidencias.

Entrega. Esta etapa consiste en la re-compilación de cada snapshot para la generación de un *release* (componente en calidad de ser implantado en producción) y publicarlo en el repositorio de artefactos, junto con su script de instalación. Una vez realizada esta tarea, la aplicación queda disponible para ser instalada en el ambiente de producción.

Despliegue. Esta etapa tiene como objetivo instalar cada release en el ambiente de producción. Para ello, se ejecuta el script entregado junto con el código, instalando de manera automática el release en el mencionado ambiente, dejando la aplicación disponible para su utilización.

Monitoreo. Esta etapa tiene como objetivo detectar errores en el software de manera temprana a través de la observación constante y automática del log de las aplicaciones en producción. De esta etapa se pueden levantar incidencias para que sean trabajadas posteriormente por los equipos de desarrollo.

Retiro. Esta es la etapa final del ciclo de vida de las aplicaciones. Tiene como objetivo eliminar el release del ambiente de producción, ya sea porque su uso ya no es necesario o porque es reemplazado por una versión más nueva. Para ello se da de baja (desinstala) del ambiente de producción.

Asimilación del pipeline por los proyectos

Las etapas descritas anteriormente son genéricas, y su utilización será por todos los proyectos del DCC. Como se explicó anteriormente, la entrada principal del pipeline son las ideas, necesidades e incidencias aportadas o reportadas por usuarios o interesados. Más de una idea podría ser desarrollada en paralelo, y, a su vez, una idea podría involucrar la creación de más de un proyecto. Al tener una capacidad limitada de desarrollo y aún muchas áreas operacionales del DCC que necesitan soporte tecnológico, la ejecución de los proyectos no es tan rápida como la generación de ideas o reporte de incidencias.

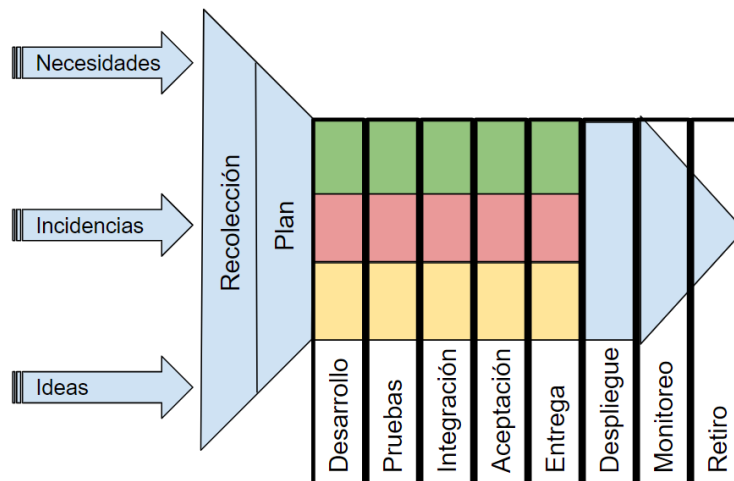


Figura 16. Utilización del pipeline por lo proyectos.

En la Figura 16, los colores representan distintos proyectos que están siendo desarrollados a la vez. Como

se puede observar, los proyectos siguen las etapas del pipeline, desde desarrollo hasta entrega. La etapa de despliegue no es de responsabilidad de los proyectos ya que es el Área Aplicaciones quien decidirá si el desarrollo es puesto en producción o no. Es importante señalar en este punto que existe una excepción para el curso Ingeniería de Software II, ya que si bien es cierto que deberán utilizar el pipeline, la metodología de desarrollo es distinta a la propuesta por lo que la interacción con el pipeline será desde la etapa de Recolección y Plan y luego se retomará el pipeline en la fase de Entrega hasta Retiro, como se muestra en la Figura 17.



Figura 17. Utilización propia del pipeline por Ingeniería de Software II.

4.1.2. Roles, personas o equipos que interactúan con el pipeline

Existen distintos actores que participan en el proceso de desarrollo de software definido. Estos tienen a su vez diferentes responsabilidades en la entrega de las aplicaciones. Los principales actores son Área Sistemas, Área Aplicaciones, Equipos de desarrollo, Usuarios y la Dirección del DCC. La caracterización de estos roles es la definida en la Sección 3.3.3. Su participación en el pipeline se define a continuación.

- *Área Sistemas.* El Área Sistemas está a cargo de construir y mantener operativos los diferentes ambientes de trabajo (desarrollo, pruebas, etc.) y las herramientas que asisten al pipeline (repositorio de artefactos, etc.). Además, el área monitorea el estado de la infraestructura y de las aplicaciones en el ambiente de producción, por lo que puede capturar y registrar incidencias (etapa de recolección).
- *Área Aplicaciones.* El Área Aplicaciones es responsable de la recolección, planificación y gestión de los proyectos en ejecución, del paso a producción (despliegue) y de la gobernanza de las aplicaciones en el ambiente de producción. Dado que el área es responsable del pipeline, está a cargo de su documentación, difusión, inducción y mejora, así como de la búsqueda de nuevas soluciones tecnológicas que asistan al pipeline.
- *Equipos de desarrollo.* Los equipos de desarrollo participan del pipeline en las etapas de construcción, desde el desarrollo hasta la entrega.
- *Usuarios.* Los usuarios participan en la etapa de recolección, levantando incidencias, estableciendo sus necesidades y proponiendo ideas de mejora a las aplicaciones. Los usuarios además participan del refinamiento de requisitos durante la etapa de desarrollo, y durante la etapa de pruebas y aceptación para validar las soluciones construidas. Los usuarios utilizan las aplicaciones en el ambiente de producción, y pueden reportar incidencias resultantes de su uso.
- *Dirección del DCC.* La Dirección del DCC participa del pipeline principalmente en la etapa de planificación para definir las prioridades del backlog. A su vez, la Dirección puede utilizar las herramientas de apoyo para monitorear el estado de progreso y salud de los proyectos y las aplicaciones.

4.1.3. Artefactos

Los artefactos son todos aquellos documentos, configuraciones o datos que son producidos y utilizados durante el desarrollo de aplicaciones. Estos artefactos son capturados, editados y archivados en herramientas de apoyo.

Tablero Kanban

El proceso definido utiliza tableros Kanban para la gestión del trabajo, tanto a nivel del Área Aplicaciones como a nivel de cada proyecto. Un tablero Kanban [15] es una herramienta que permite hacer visible el trabajo desarrollado a través del conocimiento intangible. La representación de este trabajo se realiza generalmente a través de tarjetas las cuales van avanzando por una serie de columnas definidas por el dueño del tablero, representando un sistema de flujo. Con el fin de organizar la cantidad de trabajo, se crean políticas de uso, entre las cuales está la delimitación de la cantidad de trabajo (WiP acrónimo del inglés *Work in Progress*) a un número máximo de tareas para abordar. Además, los sistemas Kanban deben tener puntos de compromiso (*Commitment*) y entrega (*Delivery*) identificados. El objetivo de definición de estos puntos de compromiso y entrega es generar un acuerdo entre el cliente y el equipo de desarrollo en que el cliente desea y asume la entrega de la tarea y el equipo de desarrollo lo producirá y entregará el cliente. Es importante destacar que el límite de WiP aplica solo para las tareas que se encuentren entre estos dos puntos señalados.

El proceso definido para el DCC incluye dos tableros Kanban. Uno de ellos está orientado a la visualización a alto nivel de las necesidades, ideas e incidencias. El otro está orientado a la visualización del progreso de cada proyecto. A continuación, se describen los tableros.

Tablero de gestión de ideas e incidencias

Este tablero estará orientado a organizar y dar visibilidad a los actores del trabajo que se está realizando en el Área Aplicaciones. Captura las ideas e incidentes reportados, y preserva y presenta su seguimiento a medida que éstas van avanzando en el pipeline. Con el fin de distinguir aquellas ideas e incidentes que provienen de usuarios finales, de las que provienen de las Áreas Aplicaciones y Sistemas, este tablero se organiza en dos vistas distintas. Una de ellas es pública, por lo que es visible para todos los interesados. La otra vista es privada, visible solo para las Áreas Aplicaciones y Sistemas. Cada vista es una implementación diferente del mismo tablero. El propósito de esta separación es mantener simple y de alto nivel la visualización que proviene de los usuarios finales de lo contrario, registros de bajo nivel reportados por los equipos de desarrollo o del Área Sistemas estarían mezclados con ideas de usuarios. La estructura de estos tableros se muestra en la Figura 18.

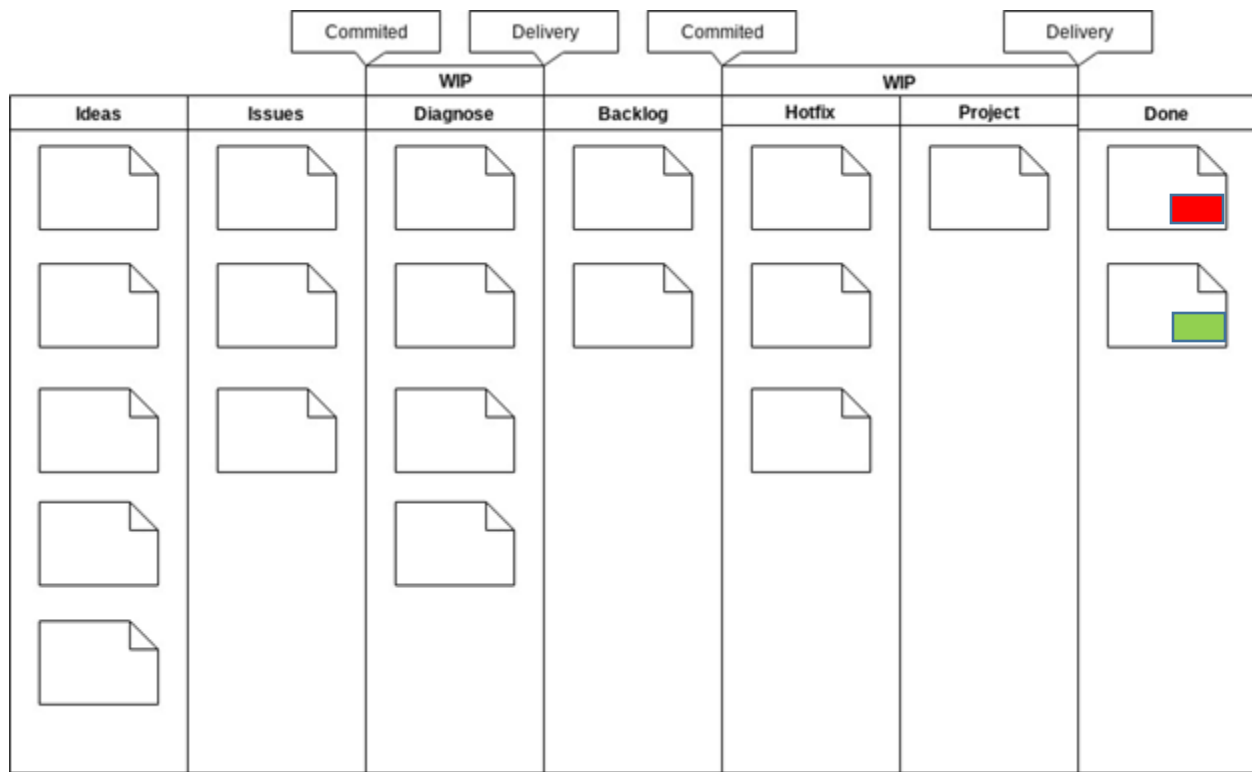


Figura 18. Tablero Kanban de ideas e incidencias.

Cada columna del tablero representa una etapa o estado por el que pasa cada idea o incidencia, existiendo una noción de progreso a medida que éstas fluyen de izquierda a derecha. Estos estados se describen a continuación:

- **Ideas.** Corresponde al backlog creado luego de la recopilación o reporte de ideas y necesidades. Las tarjetas creadas representan potenciales aportes a alguna aplicación o bien que puedan derivar en la construcción de nuevas aplicaciones.
- **Issues.** Corresponde al backlog creado luego del reporte de incidencias. Las tarjetas creadas representan errores detectados durante el uso o ejecución de las aplicaciones en el ambiente de producción.
- **Diagnose.** Las tarjetas que se encuentren en este estado son aquellas que están siendo analizadas por el Área Aplicaciones, con el fin de determinar su prioridad, criticidad y viabilidad de realización. Esta etapa tiene un WIP limitado.
- **Backlog.** Las tarjetas en este estado ya fueron analizadas y están listas para ser trabajadas como un hotfix o bien como parte de un proyecto.
- **Hotfix.** Las tarjetas en esta etapa corresponden a incidencias cuya resolución es de carácter urgente debido a estar bloqueando alguna actividad importante y necesaria en el muy corto plazo por los usuarios.
- **Project.** Las tarjetas esta etapa son aquellas que están siendo resueltas en el contexto de un proyecto.

- **Done.** Las tarjetas en esta etapa están en un estado final. Las tarjetas en este estado pueden tener las siguientes etiquetas:
 - *Complete:* esta etiqueta corresponde a las ideas o incidencias cuyo desarrollo ha finalizado y ya están disponibles para los usuarios finales.
 - *Discarded:* esta etiqueta corresponde a las ideas o incidencias que luego del análisis fueron descartadas y por lo tanto no fueron desarrolladas.

Políticas del tablero.

- Los estados iniciales son Ideas e Issues.
- El estado final es Done. Las tarjetas en este estado pueden tener las etiquetas “Complete” o “Discarded”, representadas con los colores verde y rojo, respectivamente, en la Figura 18.
- Una vez que una tarjeta pasa a Diagnose, ésta debe ser procesada ya que en esta etapa el Área Aplicaciones adquiere el compromiso de analizar la tarjeta y entregar un diagnóstico de sus implicancias.
- Existe un límite máximo de WIP para el estado Diagnose.
- El resultado del diagnóstico puede resultar en que la tarjeta pase a Backlog, o que pase a Done con la etiqueta “Discarded”.
- No hay compromiso con las tareas en la etapa Backlog. Estas tarjetas se tomarán cuando haya un equipo disponible al cual asignárselas.
- Una vez que una tarjeta se encuentra en las etapas Hotfix o Project, ésta debe ser entregada ya que se adquiere el compromiso. Entregada significa que pase a Done, ya sea como “Complete” si fue desarrollada, o “Discarded” si fue descartada durante el desarrollo.
- Existe un límite máximo de WIP para el estado Hotfix y Project.

Tablero de gestión de proyectos y hotfixes

Este tablero tiene como objetivo dar visibilidad al trabajo realizado por el equipo de desarrollo mostrándolo en términos del avance de las tarjetas en el tablero. Es importante señalar que este tablero está contenido dentro de la etapa Project o Hotfix del tablero descrito anteriormente. Con el fin de separar la resolución de incidencias y los proyectos en ejecución es que se decide separar este tablero en dos implementaciones distintas, una para cada trabajo. A diferencia del tablero anterior, la visibilidad es solo para el equipo de desarrollo.

La estructura del tablero se describe en la Figura 19.

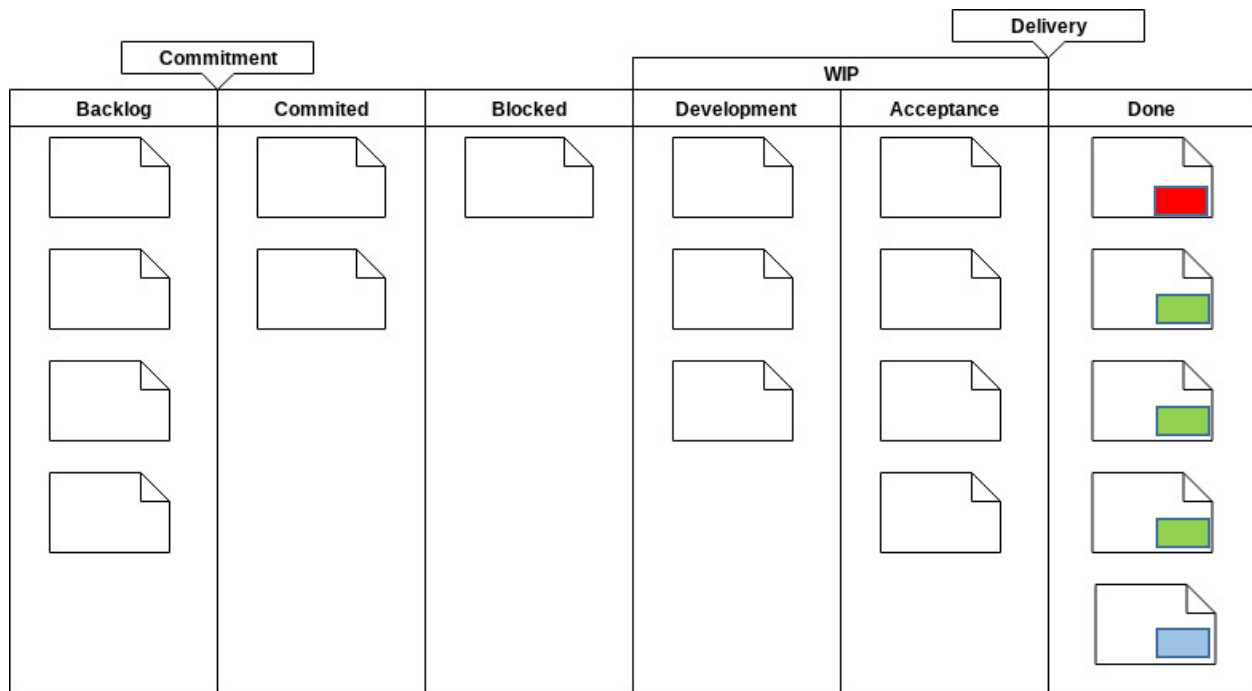


Figura 19. Tablero Kanban de proyectos y hotfixes.

Cada una de las etapas representa los siguientes estados:

- **Backlog.** Este es el punto de entrada de las tarjetas por trabajar. Luego de ser seleccionadas del Backlog del Tablero de gestión de ideas e incidencias, las tarjetas son movidas al estado Hotfix o Project, y, en consecuencia, son agregadas a la columna Backlog del Tablero de hotfixes o del Tablero del proyecto, según corresponda.
- **Committed.** Una vez haya disponibilidad por parte del equipo de desarrollo, las tareas pasarán desde la columna Backlog a esta etapa para ser analizadas por parte del equipo de desarrollo. Es posible que, debido a este análisis, pasen al estado final Done bajo la etiqueta "Discarded", o que se desagreguen en múltiples tareas.
- **Blocked.** En algunos casos, es posible que haya tarjetas que se encuentren en desarrollo y que deban ser detenidas debido a que dependen de otras actividades aun no terminadas. Estas tarjetas que se encuentran detenidas están en la columna Blocked.
- **Development.** Las tarjetas en este estado son aquellas en las cuales se está trabajando, es decir, las cuales están siendo resueltas mediante el diseño e implementación de la aplicación o los componentes correspondientes. Corresponde a esta etapa también la realización de pruebas funcionales y de integración. Para estas tarjetas, el equipo está construyendo el código fuente y realizando las pruebas necesarias. En el caso en que el equipo quede a la espera de otras tareas para poder continuar con la resolución de la tarjeta, ésta se mueve a la columna Blocked. En el caso en que el equipo decida abandonar la resolución de la tarjeta, ésta se mueve a la columna Done con la etiqueta "Cancelled".
- **Acceptance.** Las tarjetas en este estado son aquellas que superaron las tareas de integración y

por lo tanto se encuentran disponibles para ser probadas por los usuarios en pruebas de aceptación.

- **Done.** Corresponde al estado final. Este estado se puede alcanzar de distintas formas y dependiendo de cómo se llega al estado final es la etiqueta que se le dará a la tarjeta. A continuación, se describe el funcionamiento de las etiquetas:
 - *Complete.* Corresponde a la etiqueta que se asigna a las tarjetas que fueron probadas exitosamente por los usuarios, por lo que ya es posible comenzar con las tareas de armado del release para la entrega de la nueva aplicación.
 - *Cancelled.* Las tarjetas que se encuentren con esta etiqueta son aquellas que han sido comenzadas a trabajar pero que, por alguna razón, es interrumpido su desarrollo.
 - *Discarded.* Las tarjetas que se encuentren con esta etiqueta son aquellas que han sido descartadas luego de ser analizadas por el equipo de desarrollo.

Políticas del tablero.

- El estado inicial es Backlog.
- El estado final es Done y dependiendo de cómo se haya alcanzado este estado, se le dará la etiqueta "Complete", "Cancelled" o "Discarded", representadas con los colores verde, rojo y celeste en la Figura 19.
- El compromiso de entrega/finalización de la tarea se adquiere desde el estado Committed, entendiéndose por "entrega" el que la tarjeta debe llegar a un estado final.
- Existe un máximo de WIP que abarca los estados Development hasta el estado Acceptance.

Código fuente

Corresponde a la traducción de un requerimiento a un lenguaje que puede ser compilado y posteriormente ejecutado por computador.

Con el fin de conservar el código fuente desarrollado, este debe ser subido por los desarrolladores a servidores dedicados a alojar estas piezas. El modelo utilizado para el versionado y promoción del código es Gitflow Workflow [16], mientras que la numeración de la versión se realiza a través de Semantic Versioning [17].

Gitflow workflow

Es un modelo de gestión de código fuente a través de "ramas" (branches) utilizadas con distintos fines; estas son Master, Hotfix, Release, Develop y Feature. En la Figura 20 se muestra la interacción entre las distintas ramas creadas para la gestión del código fuente.

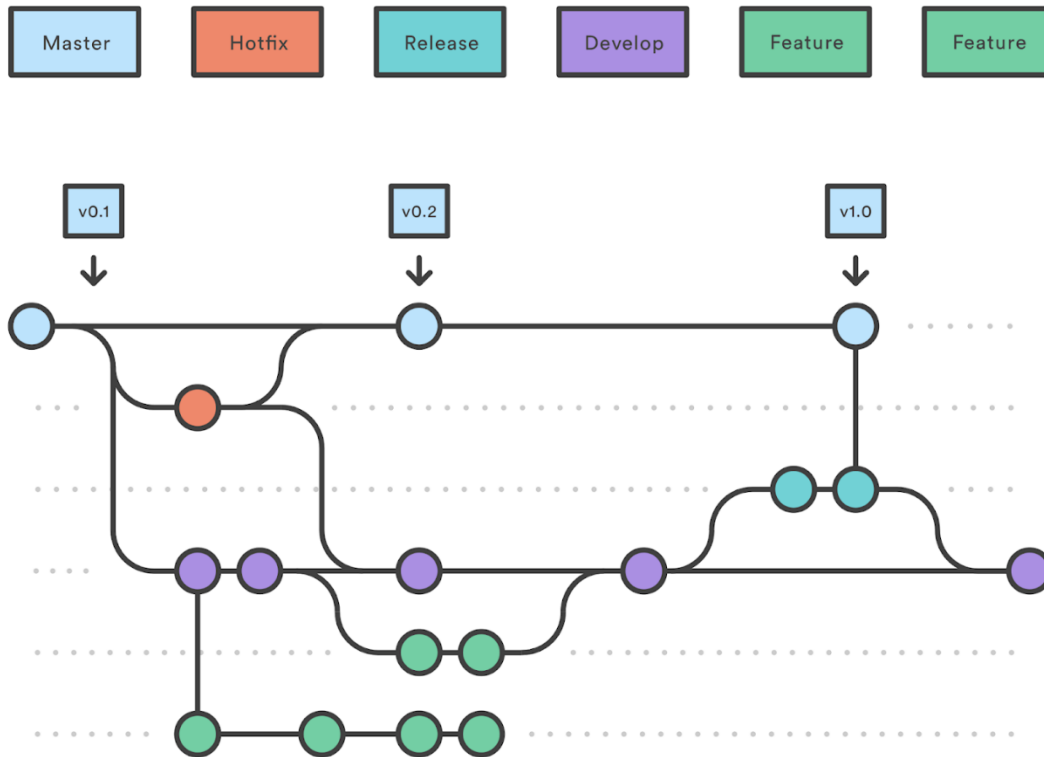


Figura 20. Manejo de branches en gitflow workflow [16].

Como se puede observar en la Figura 20, en lugar de utilizar solo una rama para desarrollar y generar releases, Gitflow Workflow propone utilizar distintas ramas para trabajar dependiendo de lo que se esté realizando (desarrollo sobre proyecto, desarrollo sobre incidencia o hotfix, desarrollo de funcionalidad o feature), posibilitando el trabajo en paralelo al hacer merge de una rama sobre otra al momento de finalizar y dejando la rama Master sólo aquello que fue liberado a producción [16].

Las aplicaciones o componentes pueden estar en calidad de snapshot o release. El desarrollo comienza generando una nueva versión en calidad de snapshot. El equipo implementa la nueva versión y realiza las pruebas correspondientes. Esto ocurre en la rama Develop y en las ramas de tipo Feature. Cuando la nueva versión está finalizada y probada por los usuarios, está queda en calidad de release y se crea la rama release correspondiente. En esta rama se ajusta la configuración de la aplicación y los scripts de instalación. Al finalizar, se realiza el merge de la rama con Master y la aplicación se implanta en producción.

Semantic versioning

Es un método de asignación de identificadores únicos a las compilaciones y versiones finales del software con el fin de organizar las dependencias hacia y desde la aplicación. El formato de la numeración es "X.Y.Z" donde X, Y y Z son números enteros no negativos y no deben tener ceros a la izquierda. X representa la versión mayor, Y representa la versión menor y Z representa la versión de parche. Cada elemento debe incrementar por separado [17], como por ejemplo:

- Pasar de 1.9.2 a 1.9.3 cuando se realiza una modificación correspondiente a mantenimiento correctivo.
- Pasar de 1.9.3 a 1.10.0 cuando se realiza una modificación aditiva, es decir, cuando el componente provee nueva funcionalidad, pero preserva los contratos (API) de la versión anterior.
- Pasar de 1.10.0 a 2.0.0 cuando se realiza una modificación no-aditiva, es decir, cuando la nueva versión altera los contratos (API) de la versión anterior.

Prueba

Corresponde a la ejecución de un escenario funcional con el fin de verificar la respuesta de la aplicación. Existen 2 tipos de pruebas presentes en el pipeline del DCC, pruebas unitarias y pruebas funcionales.

Pruebas unitarias

Son aquellas que son ejecutadas específicamente sobre la funcionalidad desarrollada o modificada. Generalmente estas pruebas son ejecutadas por el desarrollador durante la etapa Desarrollo del pipeline y pueden ser realizadas de manera manual con la aplicación implantada en un ambiente, o automática mediante la implementación del código que ejecute el escenario de prueba.

Pruebas funcionales

Son aquellas que son ejecutadas sobre escenarios funcionales que contienen a la funcionalidad desarrollada. Generalmente estas pruebas son ejecutadas por el usuario o el interesado en el sistema durante las etapas de Pruebas y Aceptación del pipeline. Estas pruebas, para el caso del DCC, no se encuentran automatizadas.

Ideas, Necesidades, Incidencias

Corresponden a las entradas del proceso de desarrollo. Éstas son recopiladas y almacenadas para su diagnóstico y potencial resolución como parte de uno o más proyectos. Es posible revisar el cambio de estado de las tarjetas que representan las ideas, necesidades e incidencias, en los tableros Kanban.

Scripts de instalación

Corresponden a rutinas que son ejecutadas para la instalación automática de las aplicaciones en los ambientes. Para la automatización de la instalación, el Área Aplicaciones del DCC construyó una herramienta que, a partir de un archivo de configuración, puede instalar las aplicaciones en cualquiera de los ambientes de la plataforma. En la Figura 21 se describe las actividades realizadas por la herramienta construida.

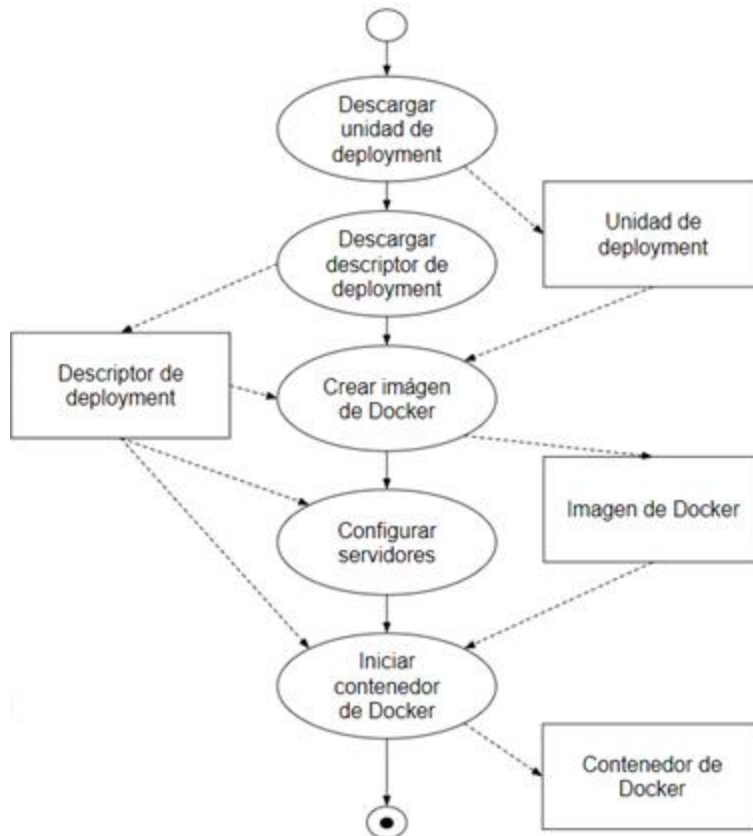


Figura 21. Tareas ejecutadas por script de instalación.

La ejecución comienza con la descarga de la *unidad de deployment* del repositorio de artefactos, la cual corresponde al ejecutable que será instalado en el ambiente. Una vez obtenida la unidad de deployment, se procede a descargar el *descriptor de deployment* desde el repositorio de configuración. Este descriptor contiene la información necesaria para que la herramienta pueda configurar el ambiente e instalar la aplicación. Específicamente, esto corresponde a la creación de una imagen de Docker, a la configuración de los servidores, y al inicio del contenedor de Docker a partir de la imagen creada.

4.2. Plataforma tecnológica

Para asistir a los interesados en el uso y ejecución del pipeline definido, se define una plataforma tecnológica que dé soporte a las etapas y actividades del pipeline. En primer lugar, y basándose en el ecosistema de herramientas de DevOps, se definió qué tipo de herramientas son necesarias. Segundo, se determinó qué herramienta concreta de cada tipo se utilizará en el DCC. El decidir los tipos de herramientas necesarios, independientemente de las herramientas concretas, permite que a futuro pueda reemplazarse una herramienta concreta por otra, preservando sí el contar con una herramienta de cada tipo.

En esta sección se documenta la plataforma tecnológica definida. Primero, se definen los tipos de

herramientas que apoyan el pipeline. Segundo, se documenta el proceso de selección de herramientas concretas, identificando los criterios de selección y buscando herramientas concretas que cumplan dichos criterios. Tercero, se define la plataforma tecnológica concreta, resultado de haber seleccionado herramientas concretas. Por último, se describen los diferentes ambientes de ejecución utilizados en el pipeline.

4.2.1. Tipos de herramientas de apoyo

Estas herramientas son tomadas desde el conjunto de herramientas del ecosistema DevOps, descritas en la Sección 2.4. En la Figura 22 se muestra la participación de estas herramientas en el pipeline definido.

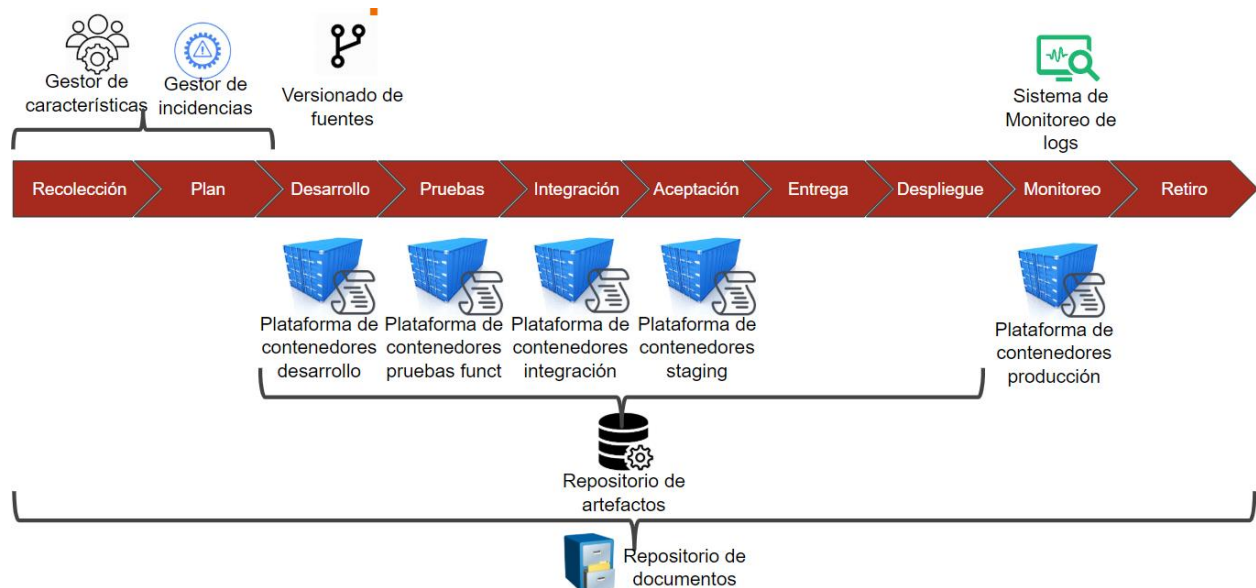


Figura 22. Herramientas de apoyo al pipeline.

En las herramientas se ejecutarán las siguientes tareas:

- **Gestor de características.** Es una herramienta que permite gestionar y visualizar el progreso de una característica o idea. En esta herramienta se implementa el tablero Kanban de gestión de ideas.
- **Gestor de incidencias.** Al igual que el gestor de características, corresponde a una herramienta que permite gestionar y visualizar el progreso de una incidencia. En esta herramienta se implementa el tablero Kanban de gestión de incidencias.
- **Versionado de fuentes.** Es una herramienta que permite la gestión de piezas de código. Por cada cambio realizado, se asigna un número de revisión. En esta herramienta se almacenan las piezas de código desarrolladas por los equipos y es controlada, como se indicó anteriormente, a través de Gitflow Workflow.
- **Repositorio de artefactos.** Es una herramienta que almacena las dependencias (bibliotecas de software de las que dependen las aplicaciones) para que sean utilizadas por los desarrolladores y

los componentes y aplicaciones implementadas para que sean instalados en los ambientes.

- **Plataforma de contenedores.** Es una herramienta que permite la virtualización a un nivel de sistema, permitiendo la existencia de múltiples espacios de usuario aislados. Con ello se aísla el sistema, evitando crear configuraciones y/o instalar programas. En esta herramienta, a través de distintas instancias, se expondrán los ambientes requeridos por el pipeline.
- **Sistema de monitoreo de logs.** Es una herramienta que permite visualizar, buscar y alertar en tiempo real en base a ciertas reglas los logs, lo cual provee un monitoreo proactivo de las aplicaciones en ejecución.
- **Repositorio de documentos.** Es una herramienta en la cual se aloja la información que describe a las aplicaciones, como las versiones y los participantes en el desarrollo, y los artefactos relevantes asociados a las aplicaciones, como documentos de requisitos, de diseño, casos de prueba, etc. En el proceso definido, se propone el uso de un wiki para el repositorio de documentos debido a su facilidad de uso y libertad en la estructura, permitiendo que con bajo esfuerzo se recopile la mayor cantidad de información sobre las aplicaciones existentes. Esto es útil en el corto plazo, pues el objetivo principal es proveer un repositorio centralizado. Sin embargo, la libertad de estructura puede llevar a que la información esté desorganizada y sea difícil de consumir. Se propone que, en el futuro, se regule el uso del wiki, dándole estructura mediante la aplicación de políticas concretas.

4.2.2. Proceso de selección de herramientas

Con el fin de obtener los criterios de selección de las herramientas necesarias, se realizaron entrevistas las Áreas Aplicaciones y Sistemas del DCC, y a empresas que cuentan o están desarrollando ecosistemas de entrega continua.

Protocolo

Los criterios de selección de herramientas se capturan mediante una entrevista semiestructurada. Al estar en un contexto de diferentes fuentes de información, pero en base al mismo objetivo de conocer los criterios necesarios para la selección de plataformas de entrega continua, es que se utiliza una pauta común para recopilar los antecedentes, dando al entrevistado como al entrevistador la libertad de expresarse sobre los temas abordados en la entrevista.

En la Tabla 4 se presenta la estructura de contenido [18] de la entrevista.

Preámbulo	Se le consulta al entrevistado si es posible grabar la conversación y se le comenta que la información obtenida en esta entrevista será utilizada en el contexto de la tesis de magíster del entrevistador y del Área Aplicaciones del DCC.
-----------	---

General	<p>Objetivos</p> <p>En el caso del DCC, el objetivo es obtener información para determinar los criterios de selección para las herramientas de desarrollo y operación que se utilizan en el departamento.</p> <p>Para el caso de la industria, el objetivo es obtener información acerca de los criterios de selección aplicados en la elección de las plataformas que utilizan, específicamente de integración continua, seguimiento de defectos (tickets), controlador de versiones y sistemas de monitoreo.</p>
Procedimiento	<p>Características de los entrevistados</p> <p>Se entrevista a los encargados de las Áreas Sistemas y Aplicaciones del DCC. En el caso de la industria, se entrevista a subgerentes de áreas informáticas y a jefes de proyectos informáticos.</p> <p>Cantidad de entrevistados</p> <p>Se entrevista a 5 personas.</p> <p>Duración de la entrevista</p> <p>Entre 15 y 30 minutos.</p> <p>Equipo y material</p> <p>Se utiliza un teléfono celular para grabar las entrevistas. Se provee al entrevistado la pauta para su análisis y se anotan en un cuaderno aspectos que podrían resultar relevantes como complemento a las grabaciones.</p>
Instrumento	<p>Guía de la entrevista</p> <p>(a) Para elegir nuevas herramientas, ¿cuáles de los siguientes criterios considera relevantes?</p> <ul style="list-style-type: none"> • Licenciamiento. • Antigüedad del producto. • Actividad de la comunidad. • Clientes principales. • Lenguajes/tecnologías soportadas. • Integración e interoperabilidad de las herramientas. • Que sea capaz de correr en los servidores del DCC (aplica solo al DCC). <p>(b) ¿Hay algún criterio que no haya sido indicado y que usted considere relevante?</p> <p>(c) Considerando los siguientes tipos de herramientas: versionado de piezas, registro de incidencias, integración continua y sistemas de monitoreo de logs,</p>

	<p>¿es posible utilizar los mismos criterios para todos ellos o existen criterios específicos que apliquen para uno o más de estos sistemas?</p> <p>(d) ¿Cuáles son las restricciones de plataforma que tienen? ¿Linux? ¿Windows?</p>
--	---

Tabla 4. Estructura de contenido entrevistas al DCC.

Ejecución

Las entrevistas fueron realizadas entre los meses de agosto y septiembre de 2017 a las Áreas Sistemas y Aplicaciones del DCC y a tres empresas que poseen o están implementando plataformas de entrega continua.

En cuanto a la caracterización de las empresas, se puede mencionar que:

- “Empresa 1” es una empresa que provee servicios de TI, entre los cuales se encuentra la implementación de sistemas de entrega continua, como también desarrollo a medida, utilizando plataformas de entrega continua. El Área de TI de la empresa mantiene y gestiona estas plataformas en las cuales se encuentra la infraestructura de sus clientes, así como también ambientes utilizados por los desarrollos internos. La empresa ya tiene implementada una plataforma de entrega continua la cual está operativa.
- “Empresa 2” es una empresa de la industria financiera. El Área TI se encarga de asegurar la interoperabilidad de los sistemas, los cuales son entregados por un proveedor de servicios con el cual llevan bastantes años y han tenido buenos resultados. La empresa está en una etapa temprana de implementación de herramientas de entrega continua.
- “Empresa 3” es una empresa que provee desarrollo de aplicaciones móviles (basadas en lenguaje Android) para la industria logística. El Área de TI está implementando un sistema de entrega continua que soporte el tipo de aplicaciones que desarrollan (móviles). Ellos no poseen área de QA, sino que liberan el software para un grupo de usuarios no informáticos los cuales realizan pruebas y crean los tickets de incidencias.

Resultados

En la Tabla 5 se pueden observar los resultados obtenidos en la pregunta (a) de la entrevista, donde una celda coloreada corresponde a una respuesta afirmativa en cuanto a la importancia del criterio de selección.

	DCC Sistemas	DCC Aplicaciones	Empresa 1	Empresa 2	Empresa 3
Licenciamiento					
Antigüedad del producto					
Actividad de la comunidad					
Interoperabilidad de las herramientas					
Tecnologías soportadas					
Clientes principales					

Tabla 5. Respuestas a la pregunta (a).

En cuanto a los resultados de la pregunta (b) acerca de si hay algún criterio que no haya sido especificado y que el entrevistado considerara relevante, las respuestas fueron las siguientes:

DCC – Sistemas:

- El Área Sistemas no indica ningún criterio adicional.

DCC – Aplicaciones:

- Que permitan maximizar la automatización.

Empresa 1:

- Que haya profesionales capacitados en el mercado.
- Que cumplan con alguna normativa, es decir, si la herramienta la utilizará alguna entidad que esté sujeta a regulaciones, esta herramienta debe cumplir con ellas.
- Que sean robustas y rápidas de aplicar.
- Que ofrezcan gobernabilidad en cuanto a saber qué es lo que hay en cada ambiente.
- Que sean de clase mundial.

Empresa 2:

- Que provean seguridad de la información.

Empresa 3:

- Que la herramienta de reporte de tickets sea fácil de usar para los usuarios finales.

En cuanto a criterios específicos para alguna de las herramientas (c), solo la Empresa 3 hizo un alcance para las herramientas de monitoreo de logs, ya que, al trabajar en el desarrollo de aplicaciones móviles

para Android, éstas son provistas por Google. En el caso de la plataforma (d), todos los participantes respondieron que utilizan Linux.

Como se puede notar, los criterios de selección del mundo de la industria son variados. Sin embargo, concuerdan en cuanto a la plataforma base, Linux, la cual coincide con el Área Sistemas del DCC. Adicionalmente a los criterios de selección, los entrevistados de la industria declararon unánimemente sentirse satisfechos con la implementación de una plataforma de entrega continua para sus aplicaciones, ayudando tanto a un ordenamiento a nivel de procesos como también a un menor *time-to-market*.

Análisis de los resultados

Para el caso del DCC, tenemos las visiones de las dos áreas que componen el mundo de DevOps, Aplicaciones (Desarrollo) y Sistemas (Operaciones). Mientras que para el Área Sistemas es importante la estabilidad de las aplicaciones más que su interoperabilidad o tecnologías soportadas, para el Área Aplicaciones son importantes criterios tales como la automatización o que las aplicaciones sean compatibles con los lenguajes o tecnologías en las cuales se desarrolla. Se hace necesario entonces obtener una visión unificada del DCC, con esto obtendremos una visión DevOps conforme a la realidad del DCC.

En el caso de las empresas, podemos observar que “Empresa 1”, dada su realidad de ser proveedores de servicios de entrega continua como también ser usuarios de esta tecnología, es importante para ellos tener criterios de selección como, por ejemplo, que haya gente capacitada en el mercado en estas plataformas, y también el hecho de que sea robusto y fácil de aplicar.

Por otro lado, para “Empresa 2”, es importante poseer herramientas que le provean seguridad de la información ya que al ser una entidad relacionada con inversiones financieras son constantemente supervisados por una superintendencia. Además, tienen un proveedor de aplicaciones con el cual llevan bastante tiempo y con el cual se sienten cómodos.

“Empresa 3” provee aplicaciones para teléfonos móviles. En su caso las restricciones están relacionadas a la usabilidad de las herramientas por parte de los usuarios finales. Es decir, Google, a través de Playstore, permite liberar la aplicación para un grupo cerrado de usuarios los cuales realizarán las pruebas de UAT y son ellos quienes deben reportar los errores. Estos usuarios no son informáticos, por lo que la herramienta de reporte debe ser amigable para estos usuarios.

Como se puede observar, existen diferencias en los criterios de selección dentro del DCC y las empresas, como también existen entre las mismas empresas. De este resultado, y limitado por la muestra estudiada, es posible concluir que los criterios de selección van ligados a la realidad de cada una de las empresas o instituciones, quienes establecen distintos niveles y prioridades a cada uno de ellos, por lo que no hay un criterio común o un estándar de criterios de selección para utilizar. Dado esto, podemos decir que los criterios de selección a utilizar pueden ser los mismos criterios establecidos por el DCC en base a su cultura o realidad, unificando lo mencionado por las Áreas Sistemas y Aplicaciones, los cuales quedan de la

siguiente manera:

- Licenciamiento.
- Antigüedad del producto.
- Actividad de la comunidad.
- Interoperabilidad de las Herramientas.
- Lenguaje / Tecnologías Soportadas.
- Herramientas que permitan maximizar la automatización.

4.2.3. Selección de herramientas

Para la selección de las herramientas que darán el soporte tecnológico al pipeline definido para el DCC, se realizó un análisis de las herramientas disponibles a la fecha de este trabajo de tesis, descartando aquellas que no cumplen con los criterios establecidos. A continuación, se listan las herramientas candidatas para cada tipo.

Repositorio de artefactos

Para el caso de herramientas que provean un repositorio de artefactos, se consultaron las fuentes [19] y [20]. La Tabla 6 muestra las herramientas candidatas, resultantes del análisis de cuáles herramientas cumplen con los criterios de selección del DCC.

Herramienta	Licenciamiento	Antigüedad del Producto	Actividad de la Comunidad	Lenguaje y Tecnologías Soportadas	Herramientas que permitan maximizar la automatización	Plataforma
Archiva	Apache Licence, V2.0	2012	Ultimo release: 2.2.3 2017-05-16. Cuenta con documentación en línea y posibilidad de reporte de bugs	Java	Se integra con EclipseIDE y Maven	Linux
Nexus	Open Source	2010	Ultimo release: 3.6.1-02 2017-11-15. Soporte y guías en línea	Java, JavaScript, .NET, php, C, C++	Se integra con Docker, Jenkins, git, IntelliJ, Eclipse, MS-Visual Studio	Unix/Windows/OSX
Pulp	Open Source	2012	Ultimo release: 2.14 2017-11-21. Soporte y guías en línea	Depende de lo que tenga configurado, RPM, Docker, etc	Plugins para RPM, Python, Puppet y Docker	Linux

Tabla 6. Repositorio de artefactos.

Control o versionado de código fuente

Para el caso de herramientas de control, se consultó [21]. La Tabla 7 muestra el resultado del análisis de cuáles herramientas cumplen con los criterios de selección del DCC.

Herramienta	Licenciamiento	Antigüedad del Producto	Actividad de la Comunidad	Lenguaje y Tecnologías Soportadas	Herramientas que permitan maximizar la automatización	Plataforma
Subversion	Open Source	2000	Ultimo Release: 2017-08-10. Cuenta con reporte de errores, wiki, entre otros	N/A		Linux
BitBucket	Open Source para instituciones académicas y sin fines de lucro	2012	Ultimo Release: 2017-10-24. V5.5. Cuenta con ayuda, guías, reporte de incidencias, entre otros	N/A	Se integra nativamente con todos los productos Atlassian y con git	Linux/Cloud
Mercurial	Open Source	2005	Ultimo Release: 2017-11-07. V 4.4.1. Cuenta con ayuda y guías en línea	N/A		Unix, Windows, macOS

Tabla 7. Herramientas de control y versionado de código fuente.

Herramientas de compilación

Para el caso de herramientas de automatización de la compilación de código fuente, se consultaron las fuentes [22] y [23]. La Tabla 8 muestra la lista de herramientas que cumplen con los criterios de selección del DCC.

Herramienta	Licenciamiento	Antigüedad del Producto	Actividad de la Comunidad	Lenguaje y Tecnologías Soportadas	Herramientas que permitan maximizar la automatización	Plataforma
Maven	Apache Licence, V2.0	2004	Ultimo Release: 2018-07-21	C/C++, Java, Ruby, Scala	Cuenta con varios PlugIns para extender funcionalidades y soportar integraciones	Unix
Gradle	Apache Licence	2007	Ultimo Release: 2018-09-19.	Java, Groovy, Scala y otros en el roadmap.	Cuenta con varios PlugIns para extender funcionalidades	Unix
SBT	BSD licence	N/D	Ultimo Release: 2018-8-7.	Java/Scala	Cuenta con PlugIns para configuraciones	Unix

Tabla 8. Herramientas de compilación.

Herramientas de monitoreo de logs

Para el caso de las herramientas de monitoreo de logs, se consultó [20]. La Tabla 9 muestra los resultados que cumplen con los criterios del DCC.

Herramienta	Licenciamiento	Antigüedad del Producto	Actividad de la Comunidad	Lenguaje y Tecnologías Soportadas	Herramientas que permitan maximizar la automatización	Plataforma
GrayLog	GNU GPL v3	2014	Ultimo Release 2017-10-19 v2.3.2 Cuenta con documentación y ayuda en línea	N/A	N/A	Linux (RedHat y Debian)
ELK (Elastic search, Logstash, Kibana)	Apache Licence, V2.0	2010	Ultimo Release 2017-11-14 v6.0.0 Cuenta con documentación y ayuda en línea	N/A	N/A	Linux

Tabla 9. Herramientas de monitoreo de logs.

Plataforma de contenedores

Para el caso de la plataforma de contenedores, se consultó [24]. La Tabla 10 muestra los resultados que cumplen los criterios de selección del DCC.

Herramienta	Licenciamiento	Antigüedad del Producto	Actividad de la Comunidad	Lenguaje y Tecnologías Soportadas	Herramientas que permitan maximizar la automatización	Plataforma
LXC (Linux Containers)	GNU LGPL v.2.1	2008	2016-11-23 v2.0.6 Cuenta con guías en línea	Depende de lo que se quiera contener	Se integra con Jenkins y Git	Linux
Docker	Apache License 2.0	2013	2017-9-26 Cuenta con guías y soporte en línea	Depende de lo que se quiera contener	Se integra con distintos servicios como AWS, Jenkins, Puppet, Azzure entre otros	Linux FreeBSD Windows x64

Tabla 10. Plataforma de contenedores.

Control de incidencias

Para el caso de la herramienta de control de incidencia, se consultó [25]. La Tabla 10 lista los resultados de aquellas que cumplen los criterios de selección del DCC.

Herramienta	Licenciamiento	Antigüedad del Producto	Actividad de la Comunidad	Lenguaje y Tecnologías Soportadas	Herramientas que permitan maximizar la automatización	Plataforma
Redmine	GNU General Public License v2	2006	Ultimo release: 2017-10-15 v3.4.3. Cuenta con ayuda y guías en línea	N/A	Se integra con distintos sistemas de control de versiones	Unix, Linux, MacOSServer, Windows
MantisBT	GNU General Public License v2	2000	Ultimo release: 2017-12-04 v2.9.0. Cuenta con ayuda y guías en línea	N/A	Se integra con distintos sistemas de control de versiones a través de plugins	Distintas plataformas , entre ellas Linux a través de LAMP

Herramienta	Licenciamiento	Antigüedad del Producto	Actividad de la Comunidad	Lenguaje y Tecnologías Soportadas	Herramientas que permitan maximizar la automatización	Plataforma
Trello	Free (con funcionalidades limitadas)	2009	N/A	N/A	Se integra con todos los productos Atlassian y con git	Cloud

Tabla 11. Gestión de ideas e incidencias.

El resultado del estudio de las herramientas que cumplen con los criterios de selección del DCC entregó varias herramientas candidatas para cada tipo de herramienta. Este resultado fue presentado al Área Aplicaciones del DCC para que tomara la decisión final de qué herramientas concretas utilizar. Las herramientas seleccionadas son las siguientes:

- Trello: herramienta web de servicio cloud que permite la gestión de proyectos y/o tareas mediante el uso de la técnica Kanban.
- Bitbucket: herramienta que permite el control y versionado de código fuente mediante Git.
- Docker: herramienta que provee una plataforma de contenedores.
- Nexus: herramienta de almacenamiento de artefactos, ya sean dependencias o artefactos propios.
- ELK: conjunto de herramientas (Elastic - Logstash - Kibana) que permite el monitoreo de logs y levantamiento de alertas basado en reglas.
- Gradle: herramienta que permite la automatización de la compilación de aplicaciones a través de scripts groovy.
- Wiki de Bitbucket: Se utilizará esta herramienta, incluida en Bitbucket, como repositorio de documentos.

La Figura 23 ilustra qué parte del pipeline asiste cada una de las herramientas seleccionadas.

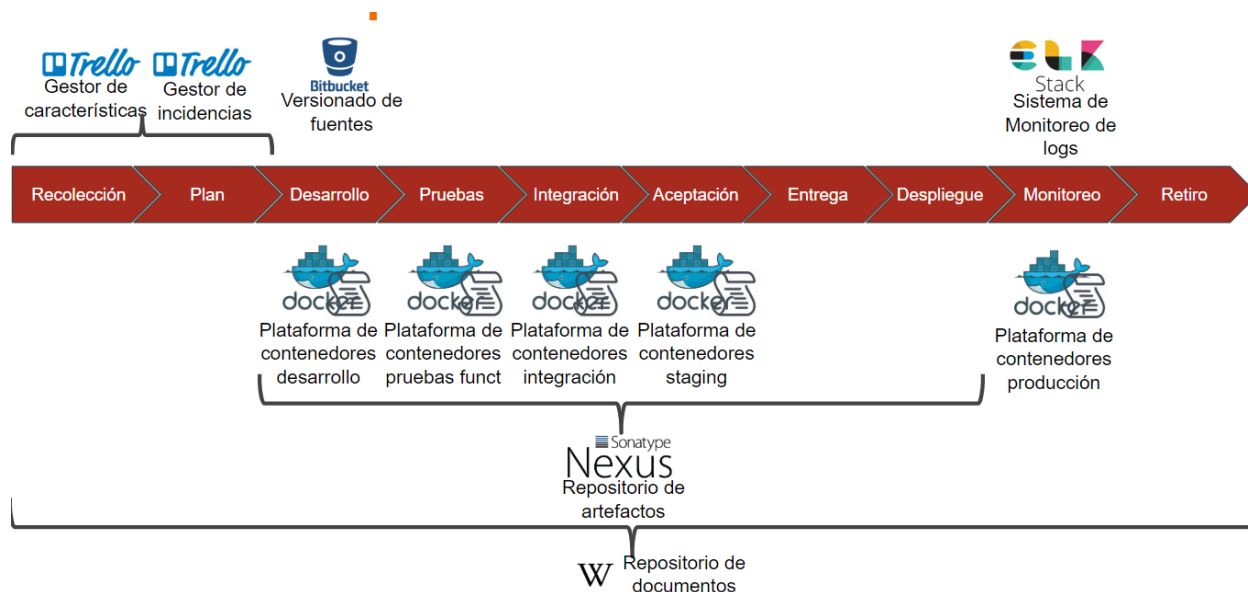


Figura 23. Pipeline con herramientas concretas.

4.2.4. Ambientes de ejecución

Los ambientes de ejecución son aquellos en los cuales se instalan las aplicaciones desarrolladas. La separación de los ambientes constituye una de las prácticas fundamentales de la disciplina de entrega continua de DevOps, tal como se describe en la Sección 2.2.2. Los ambientes de ejecución definidos para dar apoyo al pipeline del DCC son los que se muestran en la Figura 24.



Figura 24. Ambientes de apoyo al pipeline.

En particular, estos ambientes trabajan de la siguiente manera:

- *Estaciones de trabajo de los desarrolladores*: son las máquinas locales propias de cada desarrollador. En ellas se creará el código fuente de las aplicaciones. Gradle es utilizado por los desarrolladores en este ambiente para automatizar la compilación del código fuente, el empaquetado de los componentes y su eventual publicación en Nexus.
- *Ambiente de desarrollo*: corresponde a un ambiente de integración para cada proyecto. En este ambiente los desarrolladores podrán hacer pruebas funcionales menores. La instalación en este ambiente es realizada y gestionada por los desarrolladores.

- *Ambiente de pruebas*: corresponde a un ambiente con visibilidad hacia los usuarios. Acá los desarrolladores podrán instalar *snapshots* de las aplicaciones para que sean probadas por los usuarios. La instalación en este ambiente es realizada y gestionada por los desarrolladores.
- *Ambiente de integración*: una vez finalizado el desarrollo, la aplicación pasa a un ambiente de integración en la que ejecutará junto a las demás aplicaciones. De esta forma se podrán detectar errores técnicos resultantes de la interoperabilidad entre las herramientas. La instalación en este ambiente es realizada y gestionada por el Área Aplicaciones.
- *Ambiente de aceptación*: es de similares características que el ambiente de integración, es decir, la aplicación desarrollada está funcionando junto a las demás aplicaciones. La diferencia es que en este ambiente se realizan pruebas de aceptación de usuario. La instalación en este ambiente es realizada y gestionada por el Área Aplicaciones.
- *Ambiente de producción*: es aquel donde se encuentran funcionando todas las aplicaciones del DCC, estando disponibles para su uso por usuarios finales. La instalación en este ambiente es realizada y gestionada por el Área Aplicaciones.

La implementación de estos ambientes utiliza tecnología de contenedores, específicamente Docker, para tener mayor control sobre la automatización de la implantación y monitoreo, y para alinear la plataforma a la futura implementación de microservicios planificada por el Área Aplicaciones. La gestión de los hosts que proveen estos ambientes es responsabilidad del Área Sistemas.

Además de los ambientes en los que se instalan las aplicaciones desarrolladas, se define otro ambiente en el que se instalan las herramientas de apoyo al pipeline que corren on-premises. Específicamente, en este ambiente están disponibles la herramienta Nexus y el stack ELK, y su instalación está a cargo del Área Sistemas. Las herramientas Trello y Bitbucket corren en la nube, por lo que no se necesita un ambiente para ellas.

4.3. Cobertura de objetivos

En la Sección 3.3.1 se declaran los tres objetivos y motivadores principales del DCC para definir un proceso de desarrollo (Sección 4.1) y una plataforma tecnológica que le dé soporte (Sección 4.2). A continuación, se analiza qué cobertura tiene la solución construida con respecto a dichos objetivos y motivadores.

El primer objetivo o motivador es la **gobernanza sobre las aplicaciones**. Éste se logra a través de las herramientas que asisten al proceso es posible capturar información y documentar el desarrollo de software en sus dimensiones funcional y técnica. En particular, la gobernanza de las aplicaciones es posible con la implementación de:

- Un repositorio centralizado con toda la información y artefactos sobre cada aplicación (Wiki en Bitbucket).
- Un repositorio centralizado para el control y versionado del código fuente (Bitbucket).
- Un repositorio de artefactos para las unidades de instalación (deployment) de cada nueva versión

de las aplicaciones (Nexus).

- Archivos de configuración y herramientas de automatización de compilación (Gradle), publicación en Nexus (Gradle) y la instalación en los ambientes (Herramienta de automatización de la implantación desarrollada por el Área Aplicaciones). Esto hace repetible estas actividades.
- Una plataforma de contenedores que permite monitorear qué aplicaciones están instaladas y en ejecución en cada ambiente.
- Una herramienta de monitoreo de logs (ELK) que permite analizar y alertar sobre el estado de salud de las aplicaciones.

El segundo objetivo o motivador es la **estandarización de los procedimientos de desarrollo**. Éste se logra a través de la implementación de un proceso para el desarrollo de aplicaciones que abarca desde la recolección e incorporación de nuevos requerimientos o incidencias hasta el retiro de la aplicación desde producción. El pipeline provee un mecanismo para la gestión de ideas e incidencias, y el monitoreo de su avance en el desarrollo (tableros Kanban mencionados en la Sección 4.1.3). Considera también que la planificación la realiza el Área Aplicaciones y que la construcción de aplicaciones es realizada en proyectos. Para los proyectos propone un ciclo de vida iterativo incremental que incluye desde el desarrollo a la entrega (Figura 15 y Figura 16), pero permite utilizar cualquier ciclo de vida para contextos en los que iterativo incremental no sea posible (Figura 17). Sin embargo, para estos casos, establece el contrato de cómo debe entregarse la aplicación construida de forma de que sea instalable en producción (usando los archivos de configuración y la herramienta de automatización de la instalación).

El tercer objetivo o motivador es la **estandarización de los procedimientos de soporte**. Éste se logra a través de la implementación del proceso y sus herramientas es posible tener visibilidad a nivel de gestión y trazabilidad a nivel técnico de los errores ocurridos durante la utilización de las aplicaciones. A nivel de gestión, el pipeline provee una herramienta para la gestión de incidencias. A través de ella es posible hacer seguimiento del estado en el cual se encuentra y cómo fue tratada luego de ser diagnosticada, es decir, si ésta será trabajada como parte de un proyecto o como un hotfix. A nivel técnico, la incorporación de una herramienta de monitoreo automatizado de logs y la generación de alertas a través de esta herramienta apunta hacia una mantención proactiva de las aplicaciones. Por otro lado, el sistema de control de versiones deja disponible las piezas de código a los desarrolladores y les permite gestionar la creación de ramas específicas para la corrección del error en caso de ser necesario. Además, para tener mayor contexto en relación a la aplicación a intervenir y dada la alternancia de los equipos de desarrollo, se cuenta con información funcional de la aplicación en la Wiki que centraliza la información y documentación de las aplicaciones.

En la Sección 3.3 se definió el alcance y las expectativas de la plataforma de desarrollo del DCC. En particular, se enumeraron, además de los objetivos y motivadores discutidos anteriormente, las áreas funcionales, las interfaces y sistemas externos, las restricciones y los principios. En la Tabla 12 se describe qué aspecto del proceso o plataforma tecnológica definida resuelve cada uno de ellos.

Alcance	Técnica/herramienta
Áreas funcionales	
Recolección	
F.1. La plataforma provee un mecanismo de gestión de tickets	El pipeline incluye una etapa de Recolección en la que se capturan las incidencias encontradas (Sección 4.1.1). Además, provee un Tablero de gestión de ideas e incidencias que permite visualizar las incidencias y su estado de resolución (Sección 4.1.3)
F.2. La plataforma provee un mecanismo de registro y gestión de solicitud de nuevas funcionalidades.	La etapa de Recolección del pipeline captura y almacena las solicitudes de nuevas funcionalidades (Sección 4.1.1). Además, provee un Tablero de gestión de ideas e incidencias que permite visualizar las incidencias y su estado de resolución (Sección 4.1.3)
F.3. La plataforma provee un mecanismo para el registro y monitoreo del ciclo de vida de desarrollo y operación de las aplicaciones.	Es posible saber el estado de un desarrollo específico a través del Tablero de gestión de proyectos (Sección 4.1.3). A nivel de operación, la plataforma provee un sistema de monitoreo automatizado de logs de las aplicaciones operación (Sección 4.2.1).
Desarrollo	
D.1. La plataforma provee un mecanismo de versionado de código fuente.	La plataforma gestiona el código fuente utilizando GitFlow WorkFlow para el versionado y promoción de código fuente y utiliza Semantic Versioning para la identificación de compilación de los componentes (Sección 4.1.3).
D.2. La plataforma provee un ambiente de desarrollo.	La plataforma provee distintos ambientes implementados en distintas plataformas de contenedores (Sección 4.2.1). Estos ambientes a su vez son utilizados en distintas etapas del pipeline (Sección 4.2.4). La implementación de distintos ambientes provee aislamiento, seguridad (Sección 3.2.4) y es la base para alcanzar la práctica de entrega continua de DevOps.
D.3. La plataforma provee un repositorio de bibliotecas disponibles para los desarrolladores.	La plataforma incluye un repositorio de artefactos (Sección 4.2.1) para alojar ya sea sistemas propios como también gestionar la obtención de bibliotecas externas.

Alcance	Técnica/herramienta
D.4. La plataforma provee un servidor de integración de forma de asegurar que el código en el sistema de versionado pueda ser compilado para construir un ejecutable.	La plataforma provee una herramienta de gestión de piezas de código (Sección 4.2.1) que se encarga almacenar el código fuente generado por los desarrolladores. Además, se provee de una herramienta de compilación (Gradle) con el fin de crear snapshots o releases de las piezas de código de los sistemas creados o modificados (Sección 4.2.3).
D.5. La plataforma provee un ambiente para que los desarrolladores puedan administrar y que los usuarios puedan usar para realizar pruebas antes de pasar a los ambientes de aceptación.	En la etapa de pruebas del pipeline (Sección 4.2.4) se provee un ambiente implementado en una plataforma de contenedores (Sección 4.2.1).
Aceptación	
A.1. La plataforma provee un ambiente para verificar la correcta integración con otros sistemas.	El pipeline incluye una etapa de integración, en la cual se realizarán pruebas de este tipo a la aplicación en desarrollo (Sección 4.2.4.). La implementación del ambiente es a través de una plataforma de contenedores (Sección 4.2.1).
A.2. La plataforma provee un ambiente para pruebas de aceptación de usuario, de manera tal que sea posible verificar que la aplicación funcione según lo esperado.	El pipeline incluye una etapa de pruebas de aceptación (Sección 4.2.4). La implementación del ambiente que se utilizará para realizar estas pruebas es a través de una plataforma de contenedores (Sección 4.2.1).
Instalación	
I.1. La plataforma provee la capacidad de realizar instalaciones a pedido o bien automáticas en los diferentes ambientes.	El Área Aplicaciones desarrolló una aplicación que es utilizada en el pipeline para realizar instalaciones automáticas sobre algún ambiente determinado (Sección 4.1.3).
Monitoreo	
M.1. La plataforma dispara alertas al detectar errores en tiempo de ejecución de las aplicaciones instaladas.	El pipeline incluye una etapa de monitoreo en la cual el Sistema de monitoreo de logs (Sección 4.2.1) lanza alertas en caso de encontrar errores.
M.2. La plataforma provee un mecanismo que permite el registro de incidencias durante su uso.	El pipeline incluye una etapa de Recolección en la que se capturan las incidencias encontradas (Sección 4.1.1). Además, provee un Tablero de gestión de ideas e incidencias que permite visualizar las incidencias y su estado (Sección 4.1.3).

Alcance	Técnica/herramienta
Interfaces y Sistemas Externos	
S.1. Comunicación del nuevo requerimiento a través de la interfaz del sistema de tickets	Durante la etapa de Recolección, es posible registrar los nuevos requerimientos a través del tablero de gestión de ideas e incidencias (Sección 4.1.3).
S.2. Comunicación con los desarrolladores a través del sistema de versionado de código fuente.	Los desarrolladores pueden descargar las piezas de los proyectos, agregar nuevas o modificar las existentes y también dejar comentarios de los cambios hechos a las piezas de código al momento de versionarlos. Esto es posible con la Herramienta de gestión de piezas de código (Sección 4.2.1).
S.3. Comunicación con la herramienta de automatización de la implantación de las aplicaciones.	A través de la interfaz del sistema operativo, el Área Aplicaciones puede realizar instalaciones en los distintos ambientes utilizando el Script de instalación (Sección 4.1.3).
S.4. Comunicación con los administradores de plataforma y desarrolladores (si aplica) al momento de generar alertas en caso de fallas del software.	A través de la interfaz del sistema de monitoreo de logs (Sección 4.2.1), el Área Sistemas puede configurar reglas para enviar correos en caso de fallas en las aplicaciones.
Restricciones	
R.1. Los equipos de desarrollo deben seguir siendo los mismos.	Se caracterizan los roles, personas o equipos que interactúan con el pipeline (Sección 4.1.2) en la actualidad con el fin de mantenerlos.
R.2. La metodología de desarrollo no puede estar restringida.	El pipeline se amolda a las necesidades del curso o tarea que está realizando la persona que desarrolla la aplicación (Sección 4.1.1).
R.3. El equipo del Área Sistemas debe poder usar la nueva plataforma.	La selección de las herramientas que asisten a la operación incluyó los criterios del Área Sistemas (Sección 4.2.2).
Principios	
P.1. Centralización de la información.	La centralización de la información se realizó en distintas dimensiones. Primero se creó un tablero de gestión de ideas e incidencias (Sección 4.1.3) para registrar las tareas a desarrollar. Segundo, se implementó una herramienta de gestión de piezas de código (Sección 4.2.1), herramienta en la cual los desarrolladores almacenarán todo el código generado como parte del desarrollo de aplicaciones. Tercero, se implementó un

Alcance	Técnica/herramienta
	<p>repositorio de artefactos (Sección 4.2.1) donde se almacenarán los componentes, bibliotecas y dependencias necesarias para el desarrollo de aplicaciones. Finalmente, se implementó un repositorio de documentos (Sección 4.2.1) para alojar información de los proyectos.</p>
<p>P.2. Proceso soportado por herramientas colaborativas.</p>	<p>El tablero de gestión de ideas e incidencias (Sección 4.1.3) permite una interacción a través de los comentarios en las distintas ideas e incidencias. La herramienta de gestión de piezas de código (Sección 4.2.1) centraliza el desarrollo permitiendo a los desarrolladores avanzar de manera independiente, pero actualizando el mismo código. El repositorio de artefactos (Sección 4.2.1) sigue la misma línea de la herramienta de gestión de piezas de código, ya que posibilita a los desarrolladores obtener los componentes generados por otros proyectos para usarlos como dependencia. Finalmente, el repositorio de documentos permite a desarrolladores, administradores de sistemas e interesados en general obtener información de los proyectos del DCC (Sección 4.2.1).</p>
<p>P.3. Potenciar la automatización de tareas.</p>	<p>Se automatiza la instalación de las aplicaciones en los ambientes utilizando el script de instalación (Sección 4.1.3). Por otro lado, se automatiza la lectura de los archivos de log de las aplicaciones a través del sistema de monitoreo de logs (Sección 4.2.1), además se automatiza la generación de alertas en caso de ocurrir errores.</p>

Tabla 12. Verificación de completitud de alcance y expectativas.

5. Validación

La validación del proceso de gestión y desarrollo definido, y de la plataforma tecnológica que lo asiste fue realizada usando dos técnicas complementarias. La primera es la realización de un piloto, el cual evalúa la factibilidad técnica y adopción del proceso y de la plataforma en el contexto del desarrollo de una aplicación del DCC. La segunda es la medición de la satisfacción de los interesados. Para ello, se captura a través de un instrumento de medición su satisfacción y percepción respecto al proceso y plataforma y además se obtiene retroalimentación que permita una futura mejora.

5.1. Piloto

Con el objetivo de validar y evaluar la factibilidad y el uso del proceso definido y de las herramientas de apoyo, se realizó un piloto con un equipo del Área Aplicaciones del DCC. Este piloto consta de tres etapas principales.

La primera etapa es la implementación de la plataforma. Para realizarla, se solicitó apoyo al Área Sistemas para la creación de los ambientes y para la instalación de las herramientas de apoyo al pipeline. En la segunda etapa se decidió, junto al Área Aplicaciones, la aplicación que sería desarrollada siguiendo el pipeline y utilizando las herramientas de apoyo. Durante la ejecución de esta segunda etapa, se realizó una reunión semanal con el líder de proyecto evaluando en primer lugar las dificultades de uso del proceso, de la plataforma y de las herramientas de apoyo, como también el estado de avance para guiar en los esperables de cada una de las etapas del pipeline, aprovechando estas instancias para asistir a los participantes y de realizar eventuales mejoras al pipeline. En la tercera etapa, el equipo de desarrollo realizó una evaluación de retrospectiva para obtener sus impresiones para mantener, cambiar y mejorar en el uso del proceso y de las herramientas de apoyo.

5.1.1. Plataforma de Apoyo

La implementación de la plataforma comienza creando las herramientas necesarias para soportar las etapas de Recolección y Plan del pipeline. Primero se crea el *Tablero de gestión de ideas e incidencias*. Para ello, se crean en Trello dos tableros, uno que captura las ideas e incidencias provenientes de usuarios finales (público) y otro que captura aquellas provenientes de las Áreas Aplicaciones y Sistemas (interno). El tablero creado se ilustra en la Figura 25.

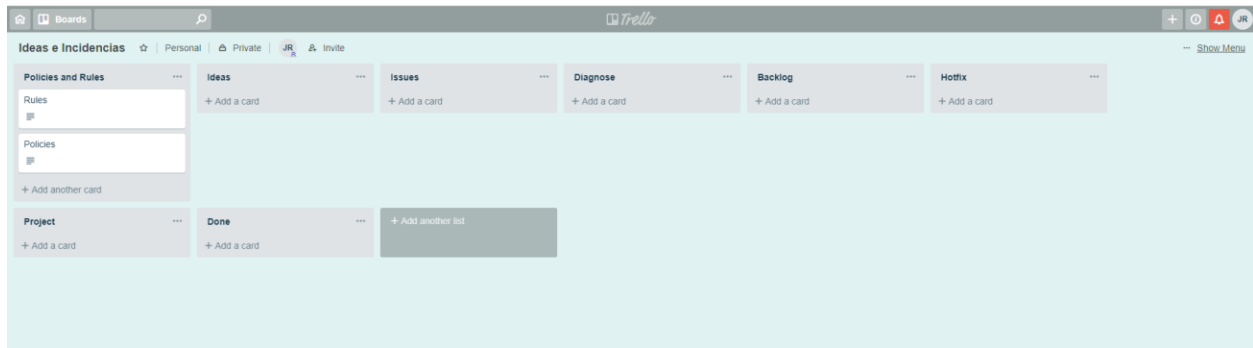


Figura 25. Tablero de gestión de ideas e incidencias en Trello.

Segundo, se crea el *Tablero de gestión de proyecto*. Tal como se explicó en la Sección 4.1.3, se utiliza un tablero por cada proyecto activo. Para este caso, no es necesario crear un tablero ya que no hay proyectos en curso. Sin embargo, aprovechando la funcionalidad de Trello de permitir clonar un tablero, se crea una *Plantilla de tablero de gestión de proyecto*, que define la estructura y que será clonada para cada proyecto que el Área Aplicaciones decida ejecutar.

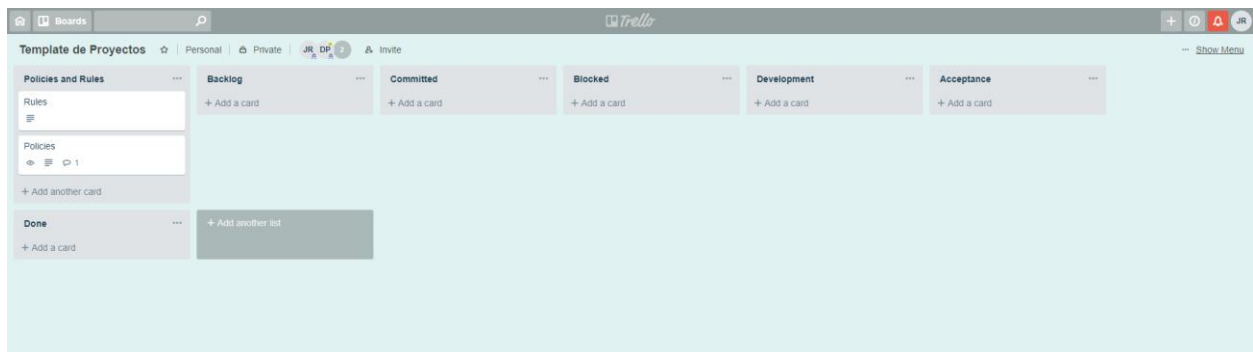


Figura 26. Plantilla de tablero de gestión de proyecto en Trello.

Finalmente, se crea el *Tablero de gestión de hotfixes* que permite gestionar el desarrollo de hotfixes de las aplicaciones de la plataforma. Este tablero es utilizado por el Área Aplicaciones para gestionar el mantenimiento correctivo de incidencias bloqueantes. Este tablero estará siempre activo y Aplicaciones dispondrá de un equipo a cargo de ejecutar las incidencias una vez reportadas. La estructura de la plantilla es igual a la del *Tablero de gestión de proyecto* ilustrado en la Figura 26.

Para la etapa de desarrollo del pipeline, la herramienta a implementar es el sistema de control de versiones, en ese contexto, cada desarrollador crea una cuenta en Bitbucket. Con el fin de ordenar el trabajo realizado, y en específico de alojar todos los repositorios que serán creados, se genera en Bitbucket un *Team* llamado *udep* al cual se agregan los miembros del Área Aplicaciones. Para agrupar los repositorios en la herramienta se utilizan los *proyectos de Bitbucket*. El Área Aplicaciones definió los siguientes grupos:

- *apps*: este grupo contiene todas aquellas aplicaciones que son del tipo web o desarrollo front-

end.

- *services*: este grupo contiene las aplicaciones del tipo back-end.
- *servers*: este grupo contiene todos los servidores que apoyan la ejecución de las aplicaciones, como, por ejemplo, servidores de base de datos, servicios de mensajería, servicio de directorio de servicios, entre otros.
- *libs*: este grupo contiene las bibliotecas compartidas por las aplicaciones.
- *docs*: es el grupo que contiene todo lo relacionado a la Wiki o repositorio de documentos.
- *tools*: es el grupo que contiene las herramientas de apoyo construidas por el DCC.

La Figura 27 muestra esta configuración inicial de Bitbucket.

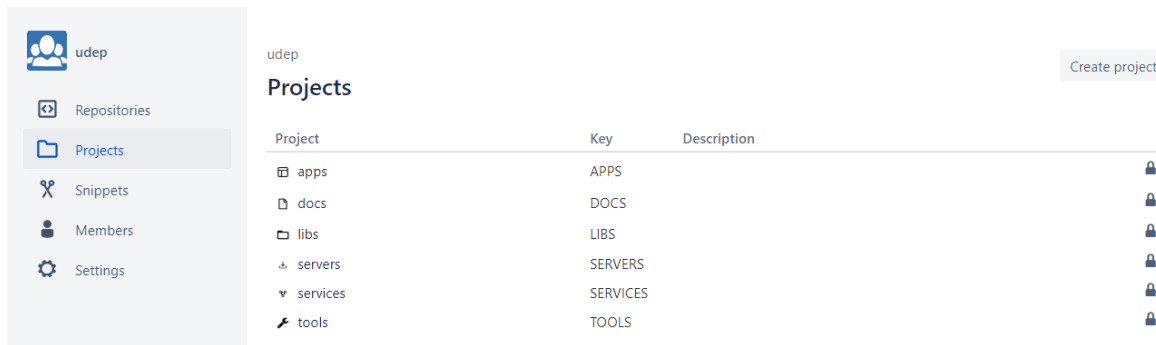


Figura 27. Configuración inicial de Bitbucket.

Con el fin de habilitar la Wiki, el Área Aplicaciones crea en el grupo docs (proyecto de Bitbucket llamado docs) un repositorio llamado “repositorio-de-documentos”. En este repositorio es creada la primera entrada en la cual está documentado el proceso de desarrollo de software, la plataforma que asiste al proceso, aspectos de arquitectura, etc. A su vez, en el wiki de este repositorio se capturará la documentación de todas las aplicaciones del DCC, registrando los participantes en el equipo de desarrollo y los artefactos asociados a cada versión.

Cabe señalar que, como se mencionó en la Sección 4.2.1, en esta etapa solo se creará el repositorio de documentos proponiendo que la gestión sea creando una página por aplicación. Sin embargo, y a medida que la plataforma vaya creciendo, se deberá realizar un trabajo de gestión documental para asegurar la disponibilidad de la información de manera eficiente y a tiempo. Esta tarea está fuera del alcance de este trabajo de tesis. En la Figura 28 se muestra una primera entrada creada por el Área Aplicaciones.



Figura 28. Wiki en Bitbucket.

La etapa de Desarrollo del pipeline necesita, además de la herramienta de control de versiones, el repositorio de artefactos para alojar los componentes creados. Se solicita al Área Sistemas la habilitación de un ambiente e instalación de Nexus en dicho ambiente. Una vez instalado, el Área Aplicaciones configura los repositorios de artefactos a utilizar. En la Figura 29 se observa la configuración realizada. Los repositorios maven-central, maven-gradle-plugins y maven-jcenter son referenciados por maven-public para facilitar las referencias desde las aplicaciones en desarrollo. Los repositorios maven-release y maven-snapshots son en los cuales se alojarán los componentes según su estado de desarrollo, vale decir, desarrollos listos para ser liberados a producción serán releases y los que se encuentran aún en desarrollo serán snapshots.

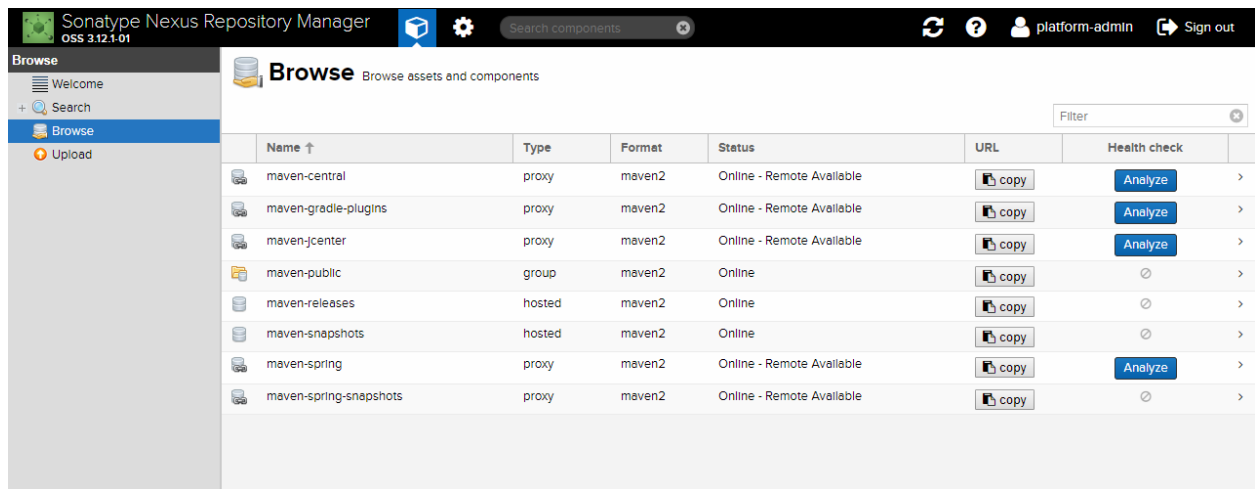
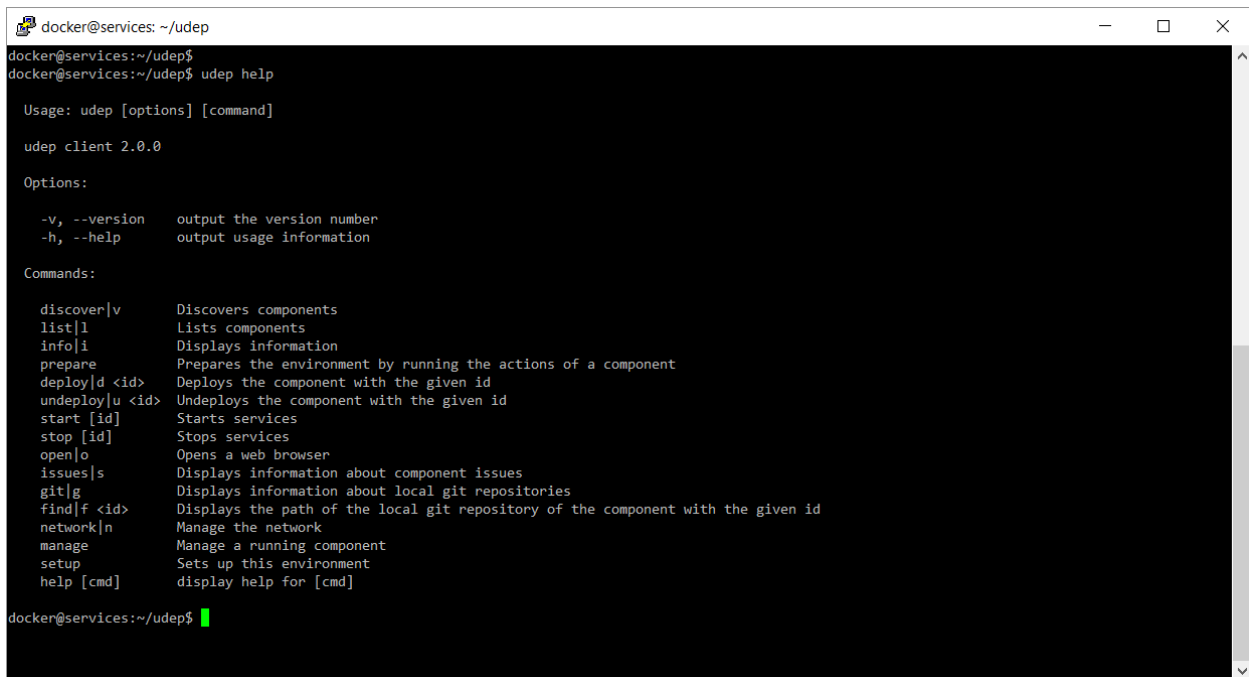


Figura 29. Repositorio de artefactos Nexus.

Para las etapas de Desarrollo, Pruebas, Integración y Monitoreo, además de las herramientas de apoyo, son necesarios ambientes que contengan en ejecución la plataforma de contenedores, tal como se describió en la Sección 4.2.4. Se solicita al Área Sistemas la habilitación de estos ambientes. Para ello, se habilitan máquinas virtuales y, dentro de ellas, se instala Docker como plataforma de contenedores para soportar las aplicaciones según lo definido en Sección 4.2.3. Si bien es cierto los ambientes se habilitan con similares características en cuanto a las herramientas instaladas en ellos, estos difieren en cuanto a capacidad, esto es, memoria y CPU. Es importante remarcar que el ambiente Staging se crea como una copia del ambiente de producción por lo que las pruebas realizadas en este ambiente tendrían resultados equivalentes a los que se obtendrían si fueran realizadas en producción. Se crea también el ambiente de Integración. Sin embargo, y debido a limitaciones en cuanto a la capacidad de la infraestructura del DCC, los ambientes de Desarrollo y Pruebas no son implementados.

Para la automatización de la instalación de los componentes en los ambientes, el Área Aplicaciones desarrolló una herramienta según lo definido en la Sección 4.1.3. Esta herramienta es instalada por el Área Sistemas en los ambientes disponibles para la ejecución del piloto. La herramienta se encuentra disponible en el repositorio de artefactos Nexus para que los desarrolladores puedan instalarla en su ambiente local. En la Figura 30 se observa una ejecución de la herramienta en la que lista sus funcionalidades.



```
docker@services: ~/udep
docker@services:~/udep$ udep help
Usage: udep [options] [command]

udep client 2.0.0

Options:
  -v, --version    output the version number
  -h, --help       output usage information

Commands:
  discover|v       Discovers components
  list|l           Lists components
  info|i          Displays information
  prepare         Prepares the environment by running the actions of a component
  deploy|d <id>   Deploys the component with the given id
  undeploy|u <id> Undeploys the component with the given id
  start |id]      Starts services
  stop |id]       Stops services
  open|o          Opens a web browser
  issues|s        Displays information about component issues
  git|g           Displays information about local git repositories
  find|f <id>     Displays the path of the local git repository of the component with the given id
  network|n       Manage the network
  manage          Manage a running component
  setup           Sets up this environment
  help [cmd]      display help for [cmd]

docker@services:~/udep$
```

Figura 30. Herramienta de automatización de la instalación.

Es importante mencionar en este punto que, al no existir sistemas en producción liberados utilizando esta forma de trabajo (pipeline y plataforma), no fue posible pilotear la herramienta de monitoreo continuo ELK, por lo que la instalación y piloto se plantea como trabajo futuro.

De esta forma, la plataforma disponible para el piloto queda como se muestra en la Figura 31, donde las herramientas que se muestran en color gris son aquellas que no fueron incluidas en el piloto.

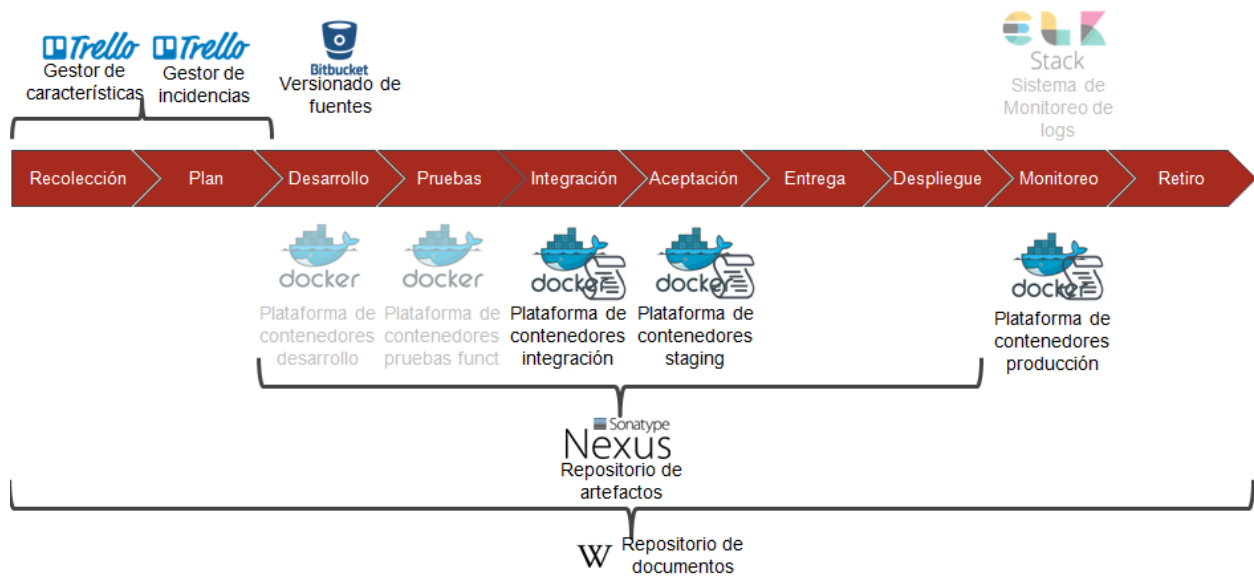


Figura 31. Plataforma implementada para el piloto.

5.1.2. Modelo de gestión

Una vez armada la plataforma, se procede a seleccionar un proyecto para la ejecución del piloto. El proyecto seleccionado es el Sistema de Gestión de la Comunidad del DCC, llamado también u-comunidad. Esta aplicación tiene como objetivo proveer al DCC un servicio centralizado con la información actualizada tanto de los miembros de la comunidad del DCC como sus roles organizacionales, permitiendo su consulta por usuarios finales y por otras aplicaciones del DCC.

La versión 1.0 de esta aplicación fue desarrollada en 2017 por un estudiante de pregrado en su memoria de grado. Sin embargo y debido a consideraciones técnicas en cuanto a la migración y utilización de nuevas arquitecturas por parte del Área Aplicaciones del DCC, esta aplicación no fue liberada a producción. Al ser un software necesario para la operación del DCC, se decide entonces hacer una refactorización de la arquitectura de la aplicación, buscando entonces construir la versión 2 de la aplicación. El piloto consiste en realizar este nuevo desarrollo.

Como primera etapa, se crea una tarjeta representativa de la necesidad en la columna Ideas del Tablero de Ideas e Incidencias, como se ilustra en la Figura 32.

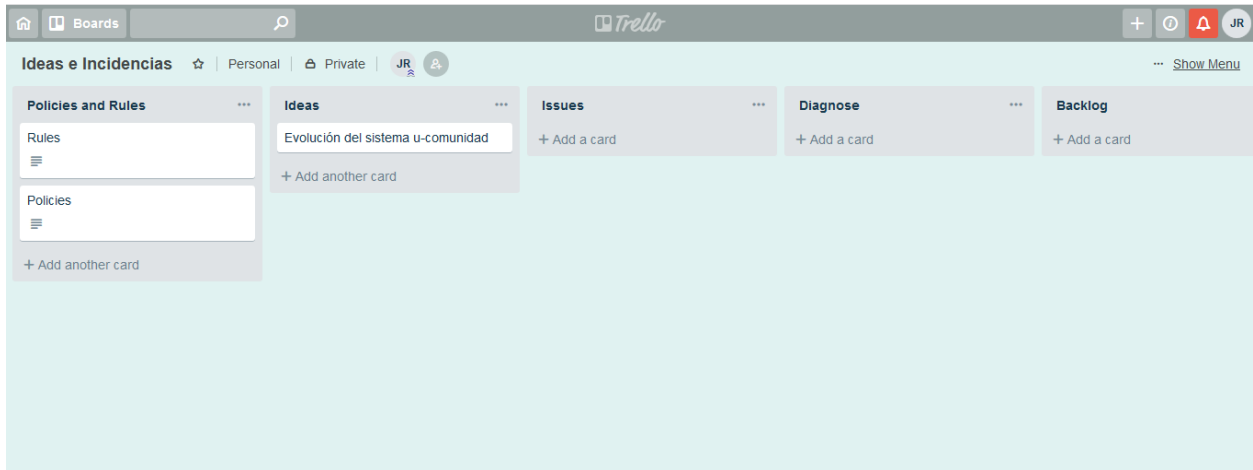


Figura 32. Registro de la idea de la evolución de u-comunidad en Trello.

Luego, la tarjeta creada se pasa a la etapa de Diagnóstico (Figura 33), donde se analiza el impacto y se evalúa el trabajo a realizar en términos de magnitud y complejidad, es decir, si es abordable como proyecto, si necesita más de un proyecto o bien si puede debe ser considerado como hotfix.

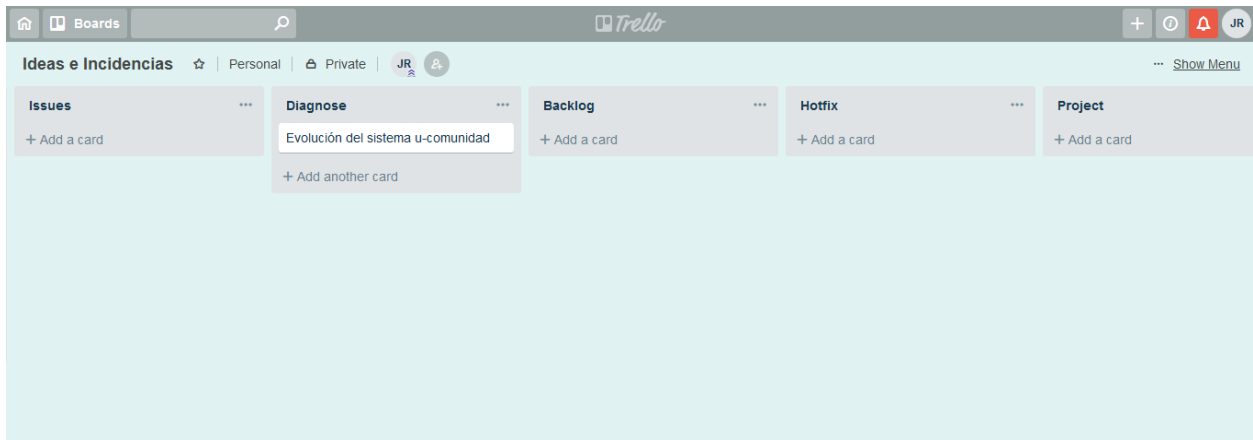


Figura 33. Idea en etapa de Diagnose del tablero.

Como parte del diagnóstico, el Área Aplicaciones recolectó toda la información y documentación relativa al sistema u-comunidad. Esto fue capturado en el Repositorio de documentos, esto es, en el Wiki de

Bitbucket llamado repositorio-de-documentos. La Figura 34 muestra la entrada en el wiki.

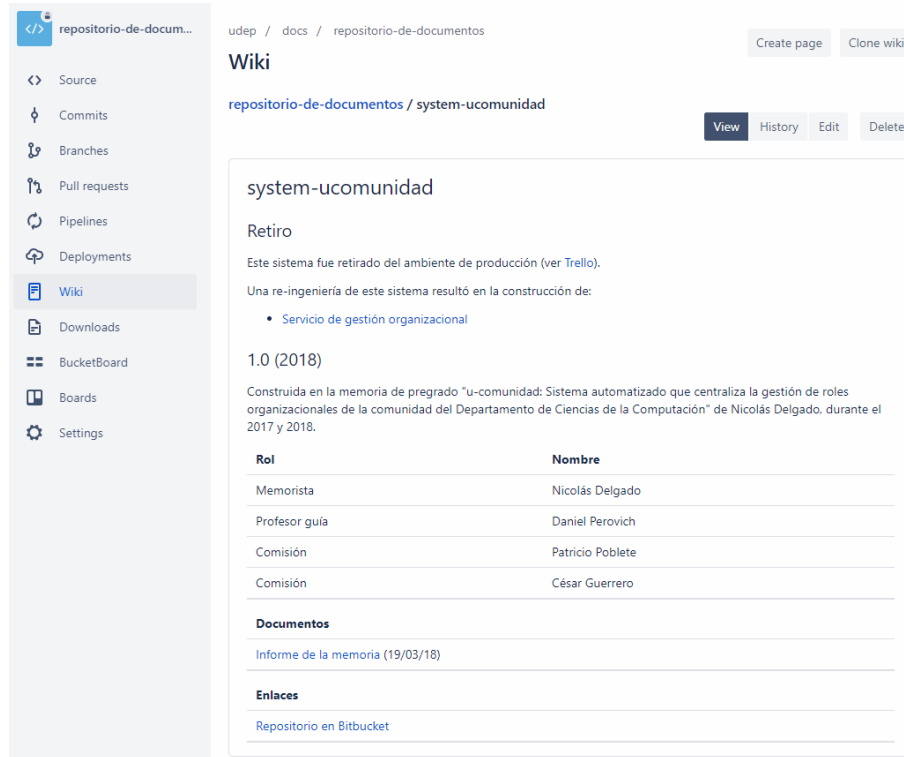


Figura 34. Wiki del proyecto U-Comunidad.

Una vez terminado el diagnóstico, la tarjeta puede pasar a estado Hotfix, Project o Done. En este caso particular, como resultado del diagnóstico realizado por el Área Aplicaciones, se crearon tres tarjetas en Backlog: App para gestión organizacional (web front-end), Servicio de gestión organizacional (back-end) y Agente de sincronización con u-campus.

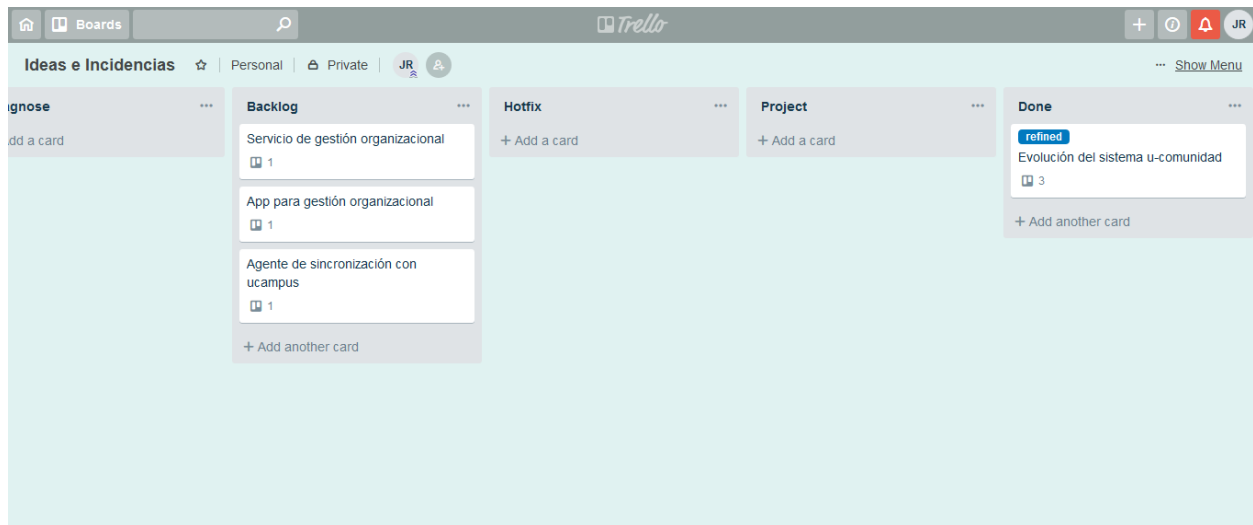


Figura 35. Nuevas tareas como resultado del diagnóstico.

Como se puede observar en la Figura 35, la tarjeta de *Evolución del sistema u-comunidad* pasó al estado Done con el label "Refined". Este label no fue definido como uno de los labels de estado final en la Sección 4.1.3, sino que surgió en el contexto del piloto como la forma de marcar las tarjetas que no fueron abordadas directamente en un proyecto o como hotfix, sino que fueron refinadas en la etapa Diagnose dando lugar a múltiples tarjetas.

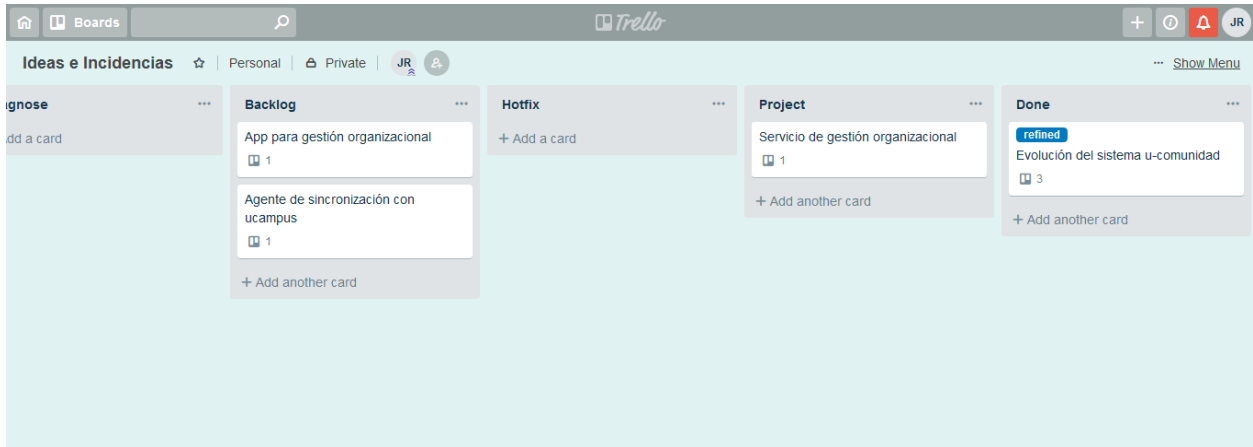


Figura 36. Tarjeta representativa del proyecto en el tablero Ideas e Incidencias.

El Área Aplicaciones decidió trabajar cada tarjeta como un proyecto independiente, comenzando por el *Servicio de gestión organizacional*. Por ello, se avanza la tarjeta al estado Project, como se muestra en la Figura 36, y se crea un tablero propio para este proyecto clonando la *Plantilla de Gestión de Proyectos*, donde además se crean tareas específicas para el desarrollo.

5.1.3. Ejecución del proyecto

El proyecto inicia con la creación de un tablero específico para su gestión. Se toma entonces como referencia el tablero *Plantilla de Gestión de Proyectos*, generando un clon según lo mencionado en la Sección 5.1.1. En este tablero se agregan al backlog las tareas que serán trabajadas en el proyecto tal como se muestra en la Figura 37.

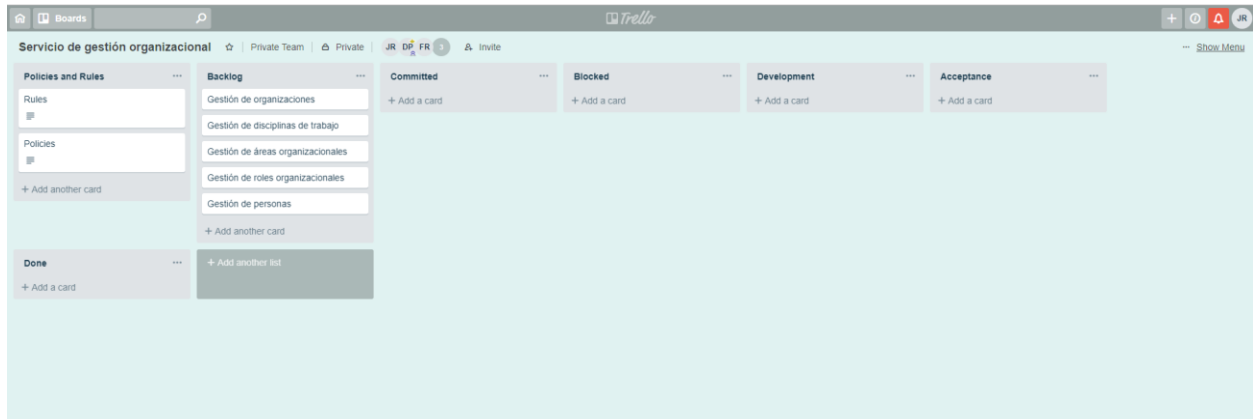


Figura 37. Tablero inicial del proyecto.

Se crea también un repositorio en Bitbucket para el proyecto como se muestra en la Figura 38. En este repositorio serán alojadas las piezas de código creadas o modificadas en el contexto de desarrollo del proyecto, así como también las diferentes ramas o branches necesarios para la implementación.

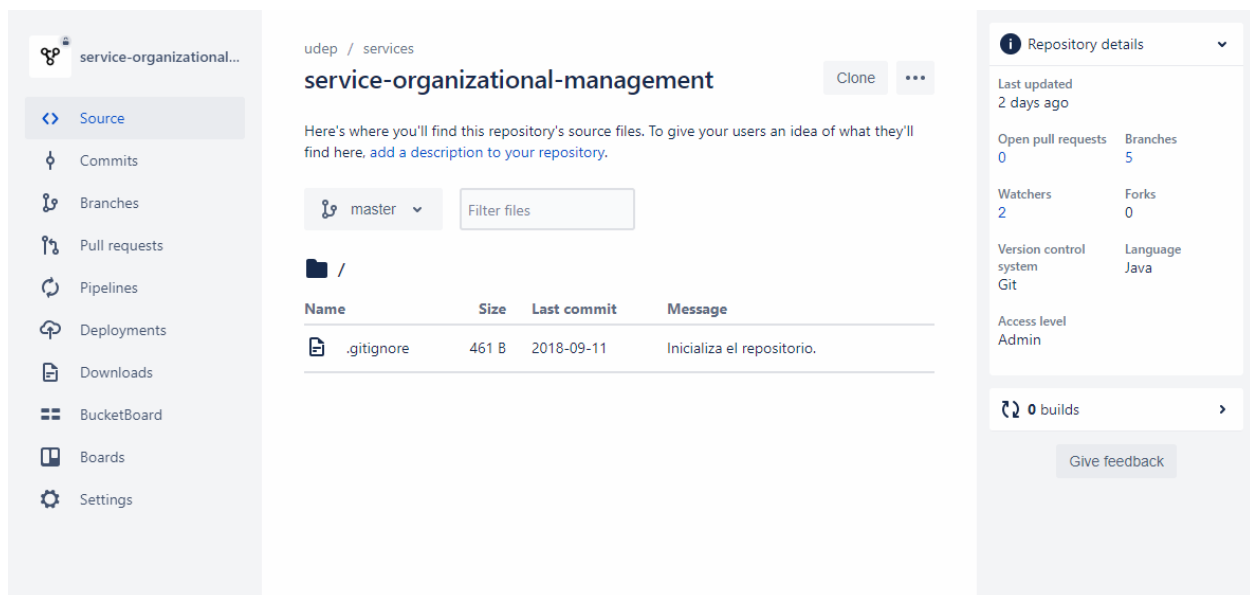


Figura 38. Espacio de versionado de fuentes para el proyecto en Bitbucket.

Una vez creado el tablero Kanban, el repositorio de código fuente, y el ambiente de Desarrollo (en las máquinas propias de los desarrolladores), el equipo de desarrollo utiliza el tablero para gestionar su trabajo. Para comenzar, se selecciona una tarjeta en el Backlog, la cual pasa a estado Committed representando que el equipo se ha comprometido a abordar dicha tarjeta. En este estado, el equipo analiza las implicancias de resolver la tarjeta, de lo cual podrían crearse nuevas tarjetas relacionadas, las cuales según su naturaleza pueden comenzar a trabajarse en forma inmediata como también podrían ser dejadas en el Backlog en espera de disponibilidad por parte del equipo. La Figura 39 muestra la evolución de las tarjetas en el proyecto.

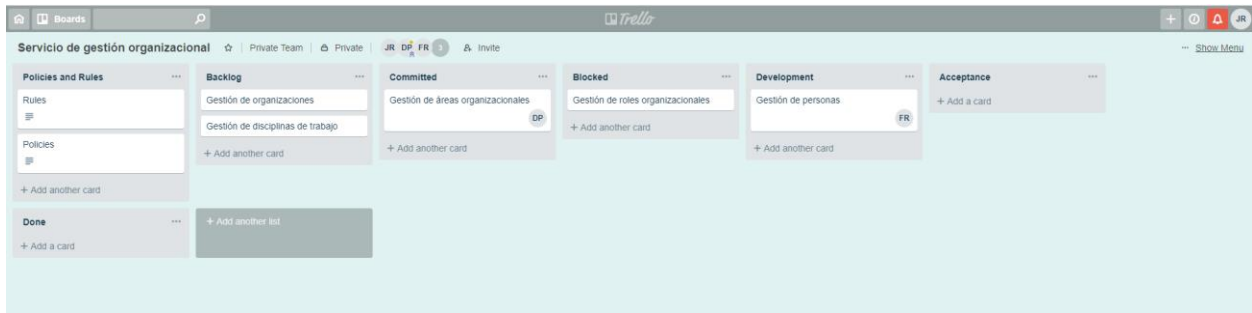


Figura 39. Tablero propio del proyecto.

A medida que son ejecutadas las tareas, comienzan a avanzar las tarjetas según sean priorizadas en el contexto del proyecto. Los desarrolladores continuarán interactuando con el sistema de control de versiones. En la Figura 40 se puede ver el repositorio de fuentes en una etapa más avanzada del proyecto.

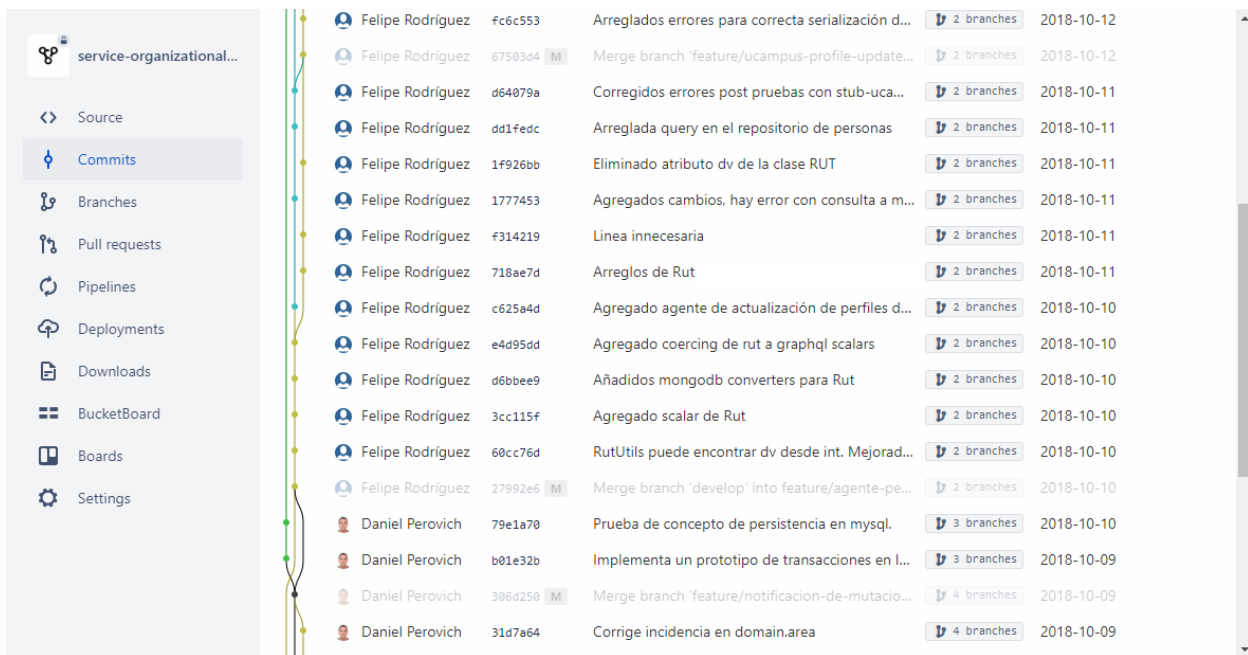


Figura 40. Evolución del versionado de fuentes.

En la Figura 41 se observa el tablero del proyecto en una etapa más avanzada. Como se puede notar, existe más de una columna que representa el estado Done. La razón de esto es que en el transcurso de la ejecución del piloto el equipo de desarrollo prefirió tener una columna Done por mes de desarrollo, con el fin de tener una mayor visualización sobre el trabajo realizado en una dimensión de tiempo.

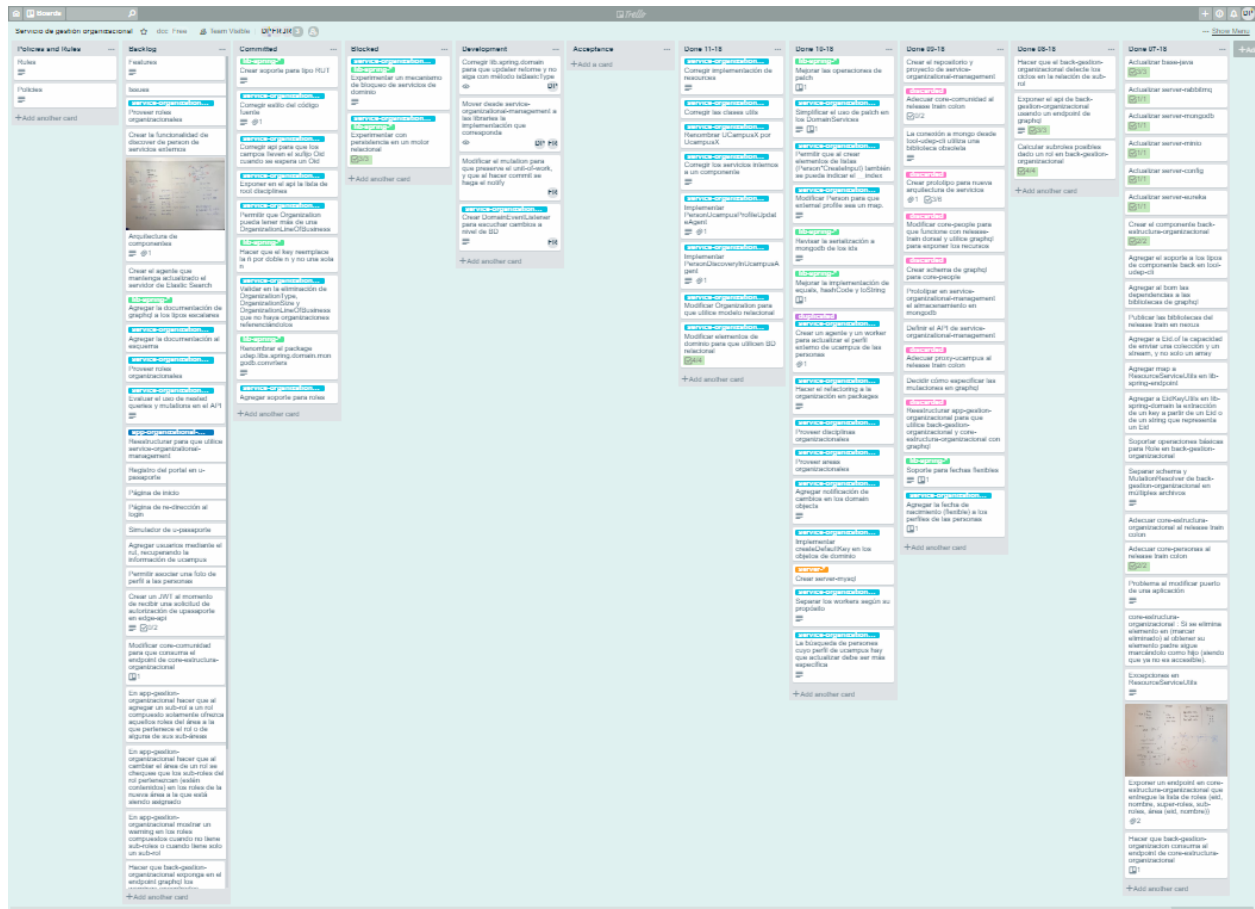


Figura 41. Tablero de proyecto en ejecución.

Una vez concluidas las actividades de desarrollo necesarias para implementar las funcionalidades comprometidas, se publica un snapshot de la aplicación en Nexus y se instala el mismo en el ambiente de Staging. La instalación en este ambiente es realizada de manera automática utilizando el script de instalación. La Figura 42 muestra el log del script de instalación, el cual registra los pasos que da la herramienta para instalar la aplicación en el ambiente. El log mostrado equivale a las acciones que realiza la herramienta, tal como se describió en la Figura 21. Por otro lado, en la Figura 43, se muestra la aplicación instalada en el contenedor del ambiente indicado.

```
docker@services: ~/udep
docker@services:~/udep$
docker@services:~/udep$ udep deploy service-organizational-management
Component to deploy: service-organizational-management:1.0.0-SNAPSHOT build 20181009.162633-6
Deploy unit downloaded
No conflicting component versions are deployed in this environment
Step 1/9 : FROM udep/base-java:1.8-1.2-SNAPSHOT
--> ccf90c83b2fe
Step 2/9 : ARG ENVIRONMENT
--> Using cache
--> 6c7063384df0
Step 3/9 : ENV ENVIRONMENT=$ENVIRONMENT
--> Using cache
--> e7fc140ba8c9
Step 4/9 : ADD udep-service-organizational-management-1.0.0-SNAPSHOT.jar app.jar
--> 950127f22615
Step 5/9 : ADD application.yml application.yml
--> 1f4b4ae6107a
Step 6/9 : RUN sh -c 'touch /app.jar /application.yml'
--> Running in 5a1d06e281b3
--> Removing intermediate container 5a1d06e281b3
--> 13f851bb720d
Step 7/9 : VOLUME /tmp
--> Running in 339190ab7ed1
--> Removing intermediate container 339190ab7ed1
--> e1485de7734f
Step 8/9 : VOLUME /logs
--> Running in 890af06b669a
--> Removing intermediate container 890af06b669a
--> 88820a952a2a
Step 9/9 : ENTRYPOINT [ "sh", "-c", "java ${javaopts} -Dspring.profiles.active=$ENVIRONMENT -Djava.security.egd=file:/dev/./urandom -jar /app.jar" ]
--> Running in 6a90e988cffc
--> Removing intermediate container 6a90e988cffc
--> 7c386ee54558
Successfully built 7c386ee54558
Successfully tagged udep/service-organizational-management:1.0.0-SNAPSHOT
Docker image created: udep/service-organizational-management:1.0.0-SNAPSHOT
Docker container created: udep-service-organizational-management-1.0.0-SNAPSHOT
Docker container started: udep-service-organizational-management-1.0.0-SNAPSHOT
docker@services:~/udep$
docker@services:~/udep$
docker@services:~/udep$
```

Figura 42. Ejecución de script de instalación.

Name	State	Created	IP Address	Published Ports
portainer	running	2018-12-16 06:59:33	192.168.0.2	9000:9000
udep-service-organizational-management-1.0.0-SNAPSHOT	running	2018-11-30 19:02:42	172.20.0.4	40531:40531
udep-server-mysql-8.0-1.0-SNAPSHOT	running	2018-10-10 09:43:23	172.20.0.3	3306:3306
udep-server-eureka-2.0-1.0-SNAPSHOT	running	2018-09-12 09:39:57	172.20.0.8	8761:8761
udep-server-config-2.0-1.0-SNAPSHOT	running	2018-08-28 16:08:59	172.20.0.9	8888:8888
udep-server-mongodb-3.6-1.2-SNAPSHOT	running	2018-08-28 15:33:46	172.20.0.10	27017:27017
udep-server-rabbitmq-3.7-1.2-SNAPSHOT	running	2018-08-28 15:29:40	172.20.0.11	5672:5672 15671:15671 15672:15672 25672:25672 4369:4369 5671:5671

Figura 43. Aplicación instalada en ambiente.

Las pruebas en el ambiente Staging son realizadas por la propia Área Aplicaciones, ya que lo que se está probando es un servicio de información, correspondiente al back-end, y no una aplicación destinada a usuarios finales. El resultado de las pruebas es la aceptación o rechazo del snapshot instalado en Staging. En caso de rechazo, se agregan nuevas tarjetas al Backlog del proyecto, representando los problemas encontrados, para que el equipo de desarrollo los resuelva. En caso de aceptación, el equipo de desarrollo crea un branch release en el que se configura la aplicación para el ambiente de Producción, y se publica

en Nexus es calidad de release. Con esto, el proyecto se da por finalizado habiendo completado el hito requerido para la fase Entrega del pipeline.

El procedimiento y decisión de instalación o liberación al ambiente de Producción es responsabilidad del Área Aplicaciones. La instalación entonces es ejecutada por esta área utilizando la herramienta de instalación en el ambiente de producción, tal como se instaló en el ambiente de Staging. Al completar este paso, la tarjeta representativa del proyecto en el Tablero de Ideas e Incidencias es movida al estado Done como se muestra en la Figura 44.

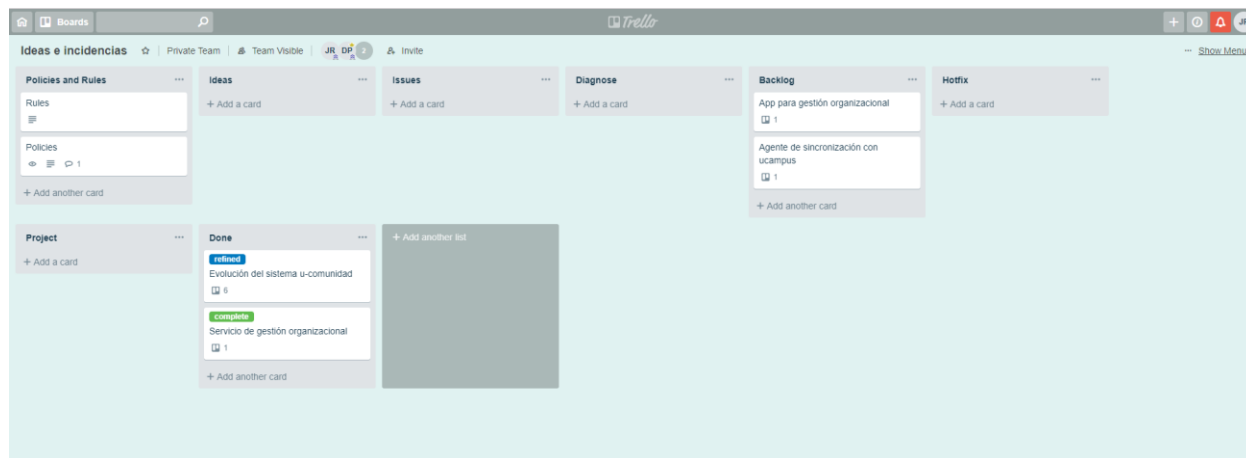


Figura 44. Servicio de gestión organizacional en estado Done con etiqueta Complete.

5.1.4. Retrospectiva

Al finalizar la ejecución del proyecto, se realizó con el equipo de una sesión de retrospectiva utilizando la técnica Keep-Fix-Try de desarrollo ágil. Además de que los participantes indiquen qué les gustaría preservar (keep), qué cosas nuevas les gustaría incluir (try), y qué les gustaría modificar (fix), deben indicar también cuáles son los riesgos que perciben del proceso y la plataforma.

Protocolo

La Tabla 13 detalla el protocolo definido para las entrevistas.

Preámbulo	Se solicita una reunión con el equipo de desarrollo a través de su líder.
General	<p>Objetivo</p> <p>Obtener información para evaluar la utilidad y usabilidad del proceso y la plataforma en el contexto de un proyecto.</p>
Procedimiento	Características de los participantes

	<p>Se entrevista al Líder del Equipo de Desarrollo del Área Aplicaciones del DCC, y al otro miembro del equipo de desarrollo.</p> <p>Cantidad de entrevistados</p> <p>Se entrevista a 2 personas.</p> <p>Duración de la entrevista</p> <p>Entre 30 y 45 minutos.</p> <p>Equipo y material</p> <p>Se utiliza una pizarra para crear los módulos y los enunciados, como también para anotar las respuestas.</p>								
<p>Instrumento</p>	<p>Guía de la actividad</p> <p>Se crean módulos para que sean respondidas las preguntas como sigue:</p> <table border="1" data-bbox="425 884 1367 1094"> <thead> <tr> <th data-bbox="425 884 660 909">Keep</th> <th data-bbox="660 884 896 909">Fix</th> <th data-bbox="896 884 1131 909">Try</th> <th data-bbox="1131 884 1367 909">Risk</th> </tr> </thead> <tbody> <tr> <td data-bbox="425 909 660 1094"></td> <td data-bbox="660 909 896 1094"></td> <td data-bbox="896 909 1131 1094"></td> <td data-bbox="1131 909 1367 1094"></td> </tr> </tbody> </table>	Keep	Fix	Try	Risk				
Keep	Fix	Try	Risk						

Tabla 13. Protocolo reunión de retrospectiva.

Ejecución

La reunión de retrospectiva se realizó en el mes de noviembre de 2018. Se realizó una sesión entre el equipo de desarrollo en la que se dieron a conocer las respuestas y comentarios acerca del uso de la plataforma y el proceso.

Resultados

La Tabla 14 presenta los resultados de la retrospectiva enumerando las respuestas entregadas por los participantes.

<p>Keep</p>	<p>K1. Uso de Semantic versioning para el versionado de componentes.</p> <p>K2. Uso de Nexus para la publicación de componentes.</p> <p>K3. Homologación de los ambientes (staging o aceptación con producción).</p> <p>K4. Automatización del deployment.</p> <p>K5. Uso de Gradle (preferido por sobre Maven con XML).</p> <p>K6. Separación de ambientes.</p> <p>K7. Habilitación del ambiente de pruebas a usuarios.</p> <p>K8. Protección del ambiente de producción.</p> <p>K9. Uso de Kanban para el seguimiento del avance del proyecto.</p>
<p>Fix</p>	<p>F1. Agregar estado “Refined” a las tarjetas del board Issues e Ideas.</p> <p>F2. Contar con la descripción del proceso y las herramientas para el equipo de desarrollo (setup del ambiente local de desarrollo).</p> <p>F3. Tener una mayor cobertura de lenguajes de programación en las herramientas de compilación (Gradle).</p> <p>F4. Oficializar el uso de la retrospectiva (se usó en el piloto, pero no está definida como parte del proceso).</p>
<p>Try</p>	<p>T1. Realizar deploy automático al hacer commit en el repositorio de control de versiones (incorporando herramientas de integración y deployment continuo como Jenkins).</p> <p>T2. Utilizar un servidor git y un servidor de wiki alternativo ya que la interfaz de usuario de Bitbucket es muy lenta.</p> <p>T3. Promover la automatización de las pruebas.</p> <p>T4. Incluir herramientas de coordinación de los equipos (como por ejemplo, Slack o Telegram).</p>

Risk	<p>R1. Sin el script de instalación, el uso de Docker es más tedioso (requiere más esfuerzo y habilidades técnicas).</p> <p>R2. Cuando se estén gestionando muchas aplicaciones, el tener un único tablero para las incidencias de todas ellas puede hacer complicado su uso.</p> <p>R3. Si no hay políticas de cómo estructurar el wiki, puede llevar a desorden.</p>
-------------	--

Tabla 14. Respuestas obtenidas en la retrospectiva.

Análisis de los resultados

Las respuestas obtenidas en la sesión de retrospectiva apoyan la decisión tomada de incluir técnicas de DevOps en el desarrollo de aplicaciones del DCC.

Primero, la separación (K6) y homologación (K3) de ambientes fue bien recibida. Es más, los participantes declaran que mediante esta técnica se resuelve un problema presente en la práctica usual del desarrollo en el DCC. Por un lado, la habilitación de un ambiente de pruebas (K7) separado del de desarrollo y del de producción permite que los usuarios finales prueben los sistemas sin bloquear el desarrollo y sin afectar los datos en producción. Por otro lado, la decisión de restringir el acceso al ambiente de Producción, de forma que sea el Área Aplicaciones la que realiza la implantación en este ambiente y no los equipos de desarrollo, permite la protección del ambiente de Producción (K8), resolviendo el problema de seguridad de dar acceso a los equipos de desarrollo a este ambiente, tal como fue explicado en la Sección 3.2.4.

Segundo, la adopción de herramientas que apoyen las diferentes etapas del pipeline de entrega fue bien recibido. El uso de una herramienta de automatización de build, y específicamente el uso de Gradle (K5) fue visto como un beneficio, y aunque el equipo sugiere una mayor cobertura de lenguajes de programación en la automatización del build (F3), hay plugins disponibles para Gradle que dan soporte a la mayoría de las tecnologías utilizadas en el DCC, reportadas en la Tabla 2. El uso de un controlador de versiones no fue mencionado directamente como un beneficio. Sin embargo, el equipo sí propone buscar un servidor alternativo a Bitbucket (T2), lo que en efecto respalda el uso de Git para este propósito. A su vez, la decisión del uso de la técnica de Semantic Versioning (K1) fue bien recibida. El equipo también encontró valor al uso de un repositorio de artefactos (K2).

Tercero, la técnica de automatización del deployment (K4) entrega valor al equipo. Incluso el equipo declara que el uso de contenedores (Docker) en los ambientes sería complejo si no se contara con la automatización (R1). Con esta técnica ya disponible, es esperable que el equipo quiera no solo automatizar el deployment, sino automatizar el inicio (trigger) del deployment al publicar (commit) nuevas versiones del código fuente en el repositorio (T1), y aún más automatizar la ejecución de pruebas (T3). La adopción de estas técnicas de DevOps se propone como trabajo futuro.

Las respuestas obtenidas permiten detectar oportunidades de mejora en el pipeline de entrega y las

herramientas de apoyo. Por un lado, el equipo encontró útil el uso de un tablero Kanban para realizar el seguimiento al proyecto (K9), sugirieron además contar con herramientas de comunicación y coordinación (T4). Por otro lado, el equipo sugirió reemplazar el servidor en Bitbucket (T2) por performance.

Algunas de las respuestas obtenidas refieren a etapas del pipeline que no están directamente involucradas con el desarrollo, lo que muestra que la forma de definir el pipeline permite al equipo tener una visión global del desarrollo, desde su concepción hasta su operación. Primero, el equipo sugirió una mejora al tablero de gestión de ideas e incidencias (F1), y alertó que su usabilidad puede disminuir al gestionar muchas aplicaciones (R2). Esto último puede resolverse utilizando varios tableros o una herramienta de gestión de incidencias (issue tracker). Analizar cuál alternativa es más apropiada para el DCC se propone como trabajo futuro. Segundo, el equipo detectó la necesidad de regular el uso del wiki para el repositorio de documentos (R3), tal como fue recomendado en la Sección 4.2.1. Es más, se recomienda utilizar el wiki también para capturar y difundir el proceso y la plataforma, de forma de que el equipo tenga acceso a este (F2).

Por último, es importante resaltar que la retrospectiva realizada en el piloto fue beneficiosa para el equipo, tan así que sugieren incluirla como parte de las actividades a realizar en el desarrollo (F4).

5.2. Validación por parte de los interesados

Con el objetivo de validar la satisfacción de los interesados con respecto al proceso y a la plataforma tecnológica de apoyo, se realizó una sesión de evaluación con los miembros del DCC que participaban y participan del desarrollo y operación de aplicaciones.

5.2.1. Protocolo

El protocolo para esta actividad consiste en dos técnicas que se ejecutan en forma consecutiva. La primera es una encuesta en escala Likert la cual se utiliza para medir la percepción de los interesados con respecto al proceso y a la plataforma. La segunda es una entrevista grupal en la que se usa como guía la encuesta para dejar que los entrevistados puedan opinar y dar sus impresiones del uso del nuevo proceso definido.

La Tabla 15 detalla el protocolo definido para las entrevistas.

Preámbulo	Se solicita una reunión con los interesados y se les expone una presentación para que puedan tener el contexto del trabajo realizado y de su implementación.
General	Objetivo Obtener información para determinar el real impacto del proceso implementado en sus interesados.

<p>Procedimiento</p>	<p>Características de los entrevistados</p> <p>Se entrevista a los encargados de las Áreas Sistemas y Aplicaciones del DCC. Además, se entrevista al académico que históricamente se ha hecho cargo del proceso de desarrollo de software en el DCC y a un integrante del equipo de desarrollo del Área Aplicaciones.</p> <p>Cantidad de entrevistados</p> <p>Se entrevista a 5 personas.</p> <p>Duración de la entrevista</p> <p>Entre 45 y 60 minutos.</p> <p>Equipo y material</p> <p>Se utiliza un teléfono celular para grabar las entrevistas. Se utiliza un computador y un proyector para realizar la presentación previa. Se provee de un ejemplar de la encuesta a cada uno de los entrevistados. Se utiliza un teléfono celular para grabar la conversación que se genera luego de la pregunta abierta.</p>																																				
<p>Instrumento</p>	<p>Guía de la entrevista</p> <table border="0" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 70%;"></th> <th style="width: 10%; text-align: center;">Mucho peor</th> <th style="width: 10%; text-align: center;">Peor</th> <th style="width: 10%; text-align: center;">Igual</th> <th style="width: 10%; text-align: center;">Mejor</th> <th style="width: 10%; text-align: center;">Mucho mejor</th> </tr> </thead> <tbody> <tr> <td>En cuanto a la gobernanza sobre las aplicaciones, cree usted que la situación actual es</td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> </tr> <tr> <td>En cuanto a la gobernanza o gestión de proyectos, cree usted que la situación actual es</td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> </tr> <tr> <td>Los procedimientos de desarrollo de software, con la implementación de la plataforma están</td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> </tr> <tr> <td>Los procedimientos de instalación de aplicaciones, con la implementación de la plataforma están</td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> </tr> <tr> <td>Los procedimientos de monitoreo y soporte, con la implementación de la plataforma están</td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> <td style="text-align: center;"><input type="checkbox"/></td> </tr> </tbody> </table>		Mucho peor	Peor	Igual	Mejor	Mucho mejor	En cuanto a la gobernanza sobre las aplicaciones, cree usted que la situación actual es	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	En cuanto a la gobernanza o gestión de proyectos, cree usted que la situación actual es	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Los procedimientos de desarrollo de software, con la implementación de la plataforma están	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Los procedimientos de instalación de aplicaciones, con la implementación de la plataforma están	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Los procedimientos de monitoreo y soporte, con la implementación de la plataforma están	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	Mucho peor	Peor	Igual	Mejor	Mucho mejor																																
En cuanto a la gobernanza sobre las aplicaciones, cree usted que la situación actual es	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																
En cuanto a la gobernanza o gestión de proyectos, cree usted que la situación actual es	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																
Los procedimientos de desarrollo de software, con la implementación de la plataforma están	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																
Los procedimientos de instalación de aplicaciones, con la implementación de la plataforma están	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																
Los procedimientos de monitoreo y soporte, con la implementación de la plataforma están	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																

	Logro de principios				
	Muy en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Muy de acuerdo
Con esta plataforma se logra centralizar la información, incluyendo código fuente e incidencias	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Con esta plataforma se instaure un proceso de desarrollo liviano	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
El soporte de herramientas de la plataforma es apropiado	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
La gestión y el control de acceso a los ambientes de producción mejoró	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Adoptabilidad de la plataforma					
En una escala de 1 a 7, donde 1 es muy malo o difícil y 7 es muy bueno o sencillo, cómo evalúa la plataforma en términos de					
	En términos de su rol		Para el DCC		
Utilidad del proceso y la plataforma	<input type="checkbox"/>		<input type="checkbox"/>		
Usabilidad del proceso y plataforma	<input type="checkbox"/>		<input type="checkbox"/>		
Capacidad de adopción o barrera de entrada	<input type="checkbox"/>		<input type="checkbox"/>		

Tabla 15. Estructura de contenido presentación del pipeline.

5.2.2. Ejecución

La sesión de validación se realizó en el mes de noviembre de 2018 con todos los interesados. La sesión comienza con la presentación del trabajo realizado dando contexto acerca de los cambios realizados y su implementación en la ejecución del piloto. Luego de finalizada la presentación, se procede a entregar la encuesta y finalmente se abre la conversación para que los participantes puedan dar su opinión y expresarse libremente acerca de las dimensiones en las cuales se evalúa cuán adoptable es la plataforma.

5.2.3. Resultados

En términos generales, los entrevistados coinciden en que la situación luego de la implementación del proceso y de la plataforma está mejor o mucho mejor que la situación anterior. Además, están de acuerdo o muy de acuerdo con que los principios de la arquitectura fueron logrados y evalúan con buenas calificaciones que tanto para ellos mismos como para el DCC esta plataforma es útil, usable y adoptable.

La Tabla 16 muestra la comparativa entre situación anterior y la situación actual.

	Sistemas 1	Sistemas 2	Aplicaciones 1	Aplicaciones 2	Aplicaciones 3
<i>Gobernanza de aplicaciones</i>	Mucho Mejor	Mejor	Mucho Mejor	Mejor	Mucho Mejor
<i>Gobernanza de proyectos</i>	Mucho Mejor	Mejor	Mucho Mejor	Mucho Mejor	Mucho Mejor
<i>Procedimiento de desarrollo</i>	Mucho Mejor	Mejor	Mejor	Mucho Mejor	Mejor
<i>Procedimiento de instalación</i>	Mucho Mejor	Mejor	Mucho Mejor	Mucho Mejor	Mejor
<i>Procedimiento de soporte</i>	Mucho Mejor	Mejor	Mucho Mejor	Mejor	Mucho Mejor

Tabla 16. Respuestas situación anterior vs situación actual.

La Tabla 17 detalla la evaluación en cuanto al logro de principios.

	Sistemas 1	Sistemas 2	Aplicaciones 1	Aplicaciones 2	Aplicaciones 3
<i>Centralizar la información</i>	Muy de acuerdo	De acuerdo	Muy de acuerdo	Muy de acuerdo	De acuerdo
<i>Proceso liviano</i>	Muy de acuerdo	De acuerdo	De acuerdo	De acuerdo	Ni de acuerdo ni en desacuerdo
<i>Soporte adecuado de herramientas</i>	Muy de acuerdo	De acuerdo	Muy de acuerdo	Muy de acuerdo	Muy de acuerdo
<i>Gestión y control de ambientes</i>	Muy de acuerdo	De acuerdo	Muy de acuerdo	Muy de acuerdo	Ni de acuerdo ni en desacuerdo

Tabla 17. Respuestas logro de principios.

La Tabla 18 muestra la calificación obtenida en cuanto a facilidad de adopción de la plataforma en términos del rol.

	Sistemas 1	Sistemas 2	Aplicaciones 1	Aplicaciones 2	Aplicaciones 3
<i>Utilidad</i>	7	6	7	7	6
<i>Usabilidad</i>	7	7	6	6,8	6
<i>Capacidad de adopción</i>	7	6	6	7	5

Tabla 18. Calificación de la plataforma en términos del rol.

La Tabla 19 muestra la calificación obtenida en cuanto a la facilidad de adopción de la plataforma en términos del dcc.

	Sistemas 1	Sistemas 2	Aplicaciones 1	Aplicaciones 2	Aplicaciones 3
<i>Utilidad</i>	7	7	7	6,8	7
<i>Usabilidad</i>	7	7	6	6	6
<i>Capacidad de adopción</i>	7	7	6	6,5	6

Tabla 19. Calificación de la plataforma en términos del DCC.

Respecto a la pregunta abierta, se recogen impresiones de los entrevistados en las siguientes frases:

- “He leído comentarios acerca de que Docker no es la mejor solución, para algunas cosas es muy bueno y para otras hay quejas”.
- “Comparado con la situación actual el uso de Docker es una mejora, ahora, si es la mejor herramienta no lo sé”.
- “(Desde el punto de vista de Ingeniería de Software II) Creo que definitivamente usar herramientas que faciliten o automaticen el trabajo es definitivamente una ganancia”.
- “(Desde el punto de vista de Ingeniería de Software II) La parte de la usabilidad no es que sea difícil usar estas herramientas, sino que en ese escenario en particular compite con las cosas que tienen que hacer en el curso más el aprender el uso de esto, aunque estoy seguro que van a estar contentos de aprender a usarlas porque es un valor agregado”.
- “Si uno mira el archivo de log de MainReq que usan los estudiantes, ellos trabajan todo por fuera y 48 horas antes suben todo al sistema”.
- “Los estudiantes trabajan en subdividir las tareas y 48 horas antes, juntar todo”.
- “En el Área Aplicaciones nos ha costado mucho mantener el tablero del proyecto al día”.
- “A veces se nos olvida (mantener el tablero del proyecto al día) y a veces terminamos teniendo

reuniones para actualizar el Kanban”

- “La sincronización que tienen (los estudiantes de Ingeniería de Software II) es caótica”.
- “Para los Memoristas y Tesistas de MTI, va a ser más fácil (adoptar el proceso y la plataforma) porque el trabajo es entre quien desarrolla y el profesor guía”.
- “Hay alguna posibilidad de flexibilizar la utilización de las herramientas, en el sentido de no restringir a los usuarios de distinto nivel a herramientas en particular”.

5.2.4. Análisis de los resultados

Como se puede observar, las Áreas Aplicaciones y Sistemas coinciden en que hubo una mejora en las dimensiones de gobernanza de aplicaciones, de proyectos, en los procedimientos de desarrollo, procedimientos de instalación y procedimientos de soporte respecto a la situación anterior luego de implementada la arquitectura y las herramientas de apoyo. Además, respecto al logro de principios de la arquitectura, en las dimensiones evaluadas también las áreas están de acuerdo en que estos son alcanzados. Finalmente, en cuanto a la utilidad, usabilidad y capacidad de adopción, las áreas también coinciden, según las calificaciones, dadas en que es una plataforma adoptable en todas estas dimensiones evaluadas, tanto para su rol como para el DCC.

Respecto a la pregunta abierta, a la luz de las frases aportadas por los entrevistados, se proponen mejoras más de forma que de fondo en cuanto a los procesos y herramientas propuestas. Por ejemplo, en la propuesta de cambiar Docker por otra herramienta de contenedores o bien en cómo hacer que los alumnos de Ingeniería de Software II puedan de alguna manera seguir el proceso y verse favorecidos por él al entrar al mundo laboral, así como también se exponen situaciones en las cuales se podría mejorar en cuanto a la actualización diaria de la herramienta de gestión de proyecto.

Es posible concluir entonces que esta plataforma es un gran avance tanto para los procesos como para la tecnología que utilizaba el DCC para el apoyo de procesos. Sin embargo, aún queda margen para mejorar, por ejemplo, dando opciones de herramientas para no depender de una única herramienta (hay una sola opción para cada herramienta de apoyo al pipeline) como bien el análisis a nivel técnico a largo plazo de las herramientas verificando que las decisiones tomadas son las mejores en esa dimensión (de largo plazo) de la realidad del DCC.

6. Conclusiones

En este capítulo se describe el trabajo realizado y se contrasta con el objetivo general propuesto para el trabajo. Luego se identifica el impacto de la solución dada primero en términos de la forma de trabajo y luego de la utilización de herramientas. Luego, se dan a conocer las lecciones aprendidas durante la realización del trabajo de tesis y finalmente se propone líneas de trabajo futuro.

6.1. Trabajo realizado

El objetivo general de este trabajo de tesis es “proveer al DCC de un modelo de gestión del desarrollo y operación de aplicaciones, basado en técnicas de DevOps, aumentando la gobernanza de la plataforma y su evolución”.

Para cumplir este objetivo, primero se investigó acerca del paradigma DevOps y del conjunto de herramientas y metodologías que son soporte de este paradigma. Luego, se identificó la forma de trabajo en cuanto al desarrollo de aplicaciones del DCC, tanto en su procedimiento como herramientas de apoyo al proceso, identificando problemas reales y potenciales debido a los procedimientos realizados. Luego, se confirmaron estas observaciones con los principales actores e interesados en el desarrollo obteniendo más información respecto a la realidad del DCC. Con estas entrevistas, se obtuvo información muy valiosa para crear una arquitectura de un modelo de gestión de desarrollo y operación de aplicaciones en el DCC.

Se utilizó la información recopilada para la definición del alcance y contexto de la solución (Sección 3.3.1), incluyendo las áreas funcionales, interfaces y sistemas externos, sistemas y datos legados, limitaciones, restricciones y principios, entre otros. En base al alcance y contexto definido, se creó un pipeline para el desarrollo de aplicaciones, identificando tareas, artefactos y roles que irán construyendo los entregables de las etapas. Además, se definió una plataforma de soporte tecnológico de apoyo al proceso con tipos de herramientas obtenidos del paradigma DevOps. Para implementar estos tipos de herramientas en herramientas concretas, se consultó a la industria, específicamente a empresas que tienen estas herramientas funcionando, y al DCC, acerca de sus criterios de selección, obteniendo como resultado que cada empresa o entidad sigue sus propios criterios. Posteriormente, se realizó un relevamiento de herramientas candidatas, se filtraron según los criterios definidos por el DCC, y se solicitó al Área Aplicaciones que seleccionara las herramientas concretas a utilizar.

Para validar el funcionamiento tanto del modelo de gestión como de las herramientas de apoyo, se realizó un piloto en conjunto con el Área Aplicaciones y el Área Sistemas. El piloto comenzó con la implementación de las herramientas de apoyo al proceso, las cuales se crearon o instalaron según el caso. Algunas herramientas quedaron fuera del piloto debido a limitaciones en los sistemas (infraestructura) del DCC o el tiempo disponible para realizarlo. El piloto continuó con la selección de una aplicación para ser desarrollada con el modelo de operación. El Área Aplicaciones seleccionó una aplicación candidata y ésta fue desarrollada siguiendo todos los pasos del pipeline, utilizando las herramientas implementadas

y generando los artefactos esperables. Para cerrar la actividad del piloto, se realizó una reunión de retrospectiva con el equipo, obteniendo puntos de satisfacción, puntos de mejora, de implementación para futuros proyectos y de riesgo.

Como una segunda etapa en la validación, se realizó una entrevista grupal a los interesados en la que se mostró la nueva forma de trabajo y se consultó las impresiones de los interesados con respecto al proceso y la plataforma.

Dado los resultados obtenidos en la validación, es posible concluir que el objetivo general de este trabajo de tesis fue logrado, proveyendo al DCC de una arquitectura que consta de un modelo de gestión de aplicaciones y de una plataforma de apoyo que está alineada con las tendencias actuales de desarrollo de software y con la realidad del DCC

6.2. Impacto de la solución

La solución construida implica un cambio profundo en el procedimiento de desarrollo de software en el DCC. Éste cambio puede ser descrito en dos dimensiones en términos de la solución dada.

El primero tiene relación con la forma de trabajo en sí misma. Si bien es cierto que previo a la implementación del pipeline existía un proceso implícito no documentado que era seguido por los actores, al llevar el desarrollo a un procedimiento explícito con artefactos entregables en las etapas, se espera que a futuro se incremente la cantidad y calidad de las aplicaciones que pase a producción.

La segunda dimensión es en cuanto a la utilización de herramientas. Las herramientas de apoyo son esenciales no solo para que el proceso funcione, sino que además funcionan como repositorios de información a futuro, ayudando a quienes busquen información acerca de los proyectos, su historia y sus piezas de software, la obtengan en su totalidad y a tiempo.

Por último, la automatización protege los ambientes y agiliza los tiempos de desarrollo quitando responsabilidades a los desarrolladores.

6.3. Lecciones aprendidas

Durante la realización de este trabajo de tesis fue posible comprobar que, tan importante como tener herramientas de apoyo y automatización es, primero, tener un proceso sobre el cuál trabajar determinando hitos o entregables en cada una de las etapas. La adopción de DevOps obliga a quién lo esté implementando a tener un procedimiento de desarrollo de software definido y documentado (pipeline), las herramientas y los artefactos van de la mano con las tareas y resultados de las etapas del pipeline. Ordenar y organizar el trabajo de desarrollo es fundamental para poder implementar herramientas de entrega continua.

Además, la creación de una arquitectura no es una tarea sencilla. Identificar los principios y a los interesados clave es tan fundamental como las restricciones y alcance. El no tener interesados con las características necesarias [14] puede llevar al fracaso en la arquitectura propuesta.

6.4. Trabajo futuro

Este trabajo de tesis representa el primer paso en la organización y gobernanza del desarrollo de software del DCC. Como trabajo futuro queda primero alcanzar la integración continua, iniciando la compilación en forma automática al momento en que los desarrolladores realizan commit de su código en el controlador de versiones. Además, y como parte de la integración continua, tener un set de pruebas unitarias que se ejecuten al momento de realizar la compilación. En una segunda etapa, avanzar hacia la entrega continua, implementando una herramienta que orqueste el proceso y que se alinee con los requerimientos del DCC, como por ejemplo Jenkins. En una tercera etapa, se podría alcanzar la implantación continua o deployment continuo, ejecutando la herramienta de instalación DCC de manera automática una vez finalizada la entrega desde Jenkins. Finalmente, y luego de haber alcanzado instalación continua se puede alcanzar el monitoreo continuo con la implementación del stack ELK.

Además, y como se mencionó en la Sección 4.1.3, se deberá hacer un trabajo de gestión documental para la Wiki de la plataforma, ya que a medida de que vaya aumentando la cantidad de aplicaciones que sean desarrolladas utilizando la plataforma, la cantidad de información irá aumentando y por lo tanto se hará necesario tener esta información organizada de manera clara para obtener la información completa y a tiempo de los proyectos.

7. Bibliografía

- [1] V. Beal, «Information Silo,» [En línea]. Available: http://www.webopedia.com/TERM/I/information_silo.html. [Último acceso: Diciembre de 2018].
- [2] M. Hüttermann, Devops for Developers, Integrate Development and Operations, the Agile Way, Apress, 2012.
- [3] G. Kim, P. Debois, J. Willis, J. Humble y J. Allspaw, The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations, IT Revolution Press, LLC., 2016.
- [4] R. Wilsenach, «DevOps Culture,» 9 7 2015. [En línea]. Available: <https://martinfowler.com/bliki/DevOpsCulture.html>. [Último acceso: Diciembre de 2018].
- [5] S. Sharma y B. Coyne, DevOps for Dummies, John Wiley & Sons, Inc., 2015.
- [6] J. Humble y D. Farley, Continuous Delivery, Reliable Software Releases through Build, Test and Deployment Automation, Addison Wesley, 2010.
- [7] L. Bass, I. Weber y L. Zhu, DevOps, A Software Architect's Perspective, Addison-Wesley, 2015.
- [8] M. Fowler, «Continuous Delivery,» 30 5 2013. [En línea]. Available: <https://martinfowler.com/bliki/ContinuousDelivery.html>. [Último acceso: Diciembre de 2018].
- [9] A. Coins, «A Generalized Model for Automated DevOps,» 2014.
- [10] S. Sharma, «Adopting DevOps for Continuous Innovation,» 2 6 2014. [En línea]. Available: <https://www.ibm.com/developerworks/library/d-devops-continuous-innovation/>. [Último acceso: Diciembre de 2018].
- [11] CA Technologies, «DevOps Perspectives 3. Straight talking and the latest thinking from the DevOps frontline,» 2015.

- [12] M. Kazuhiro y K. Takeda, «Toyota Motor Manufacturing, U.S.A., Inc.,» 1995.
- [13] M. C. Bastarrica, D. Perovich, J. Marin y R. L., «Process-Based Project Management and SPI,» 2017.
- [14] N. Rozanski y E. Woods, Software Systems Architecture, 2005.
- [15] D. J. Anderson y A. Carmichel, Essential Kanban condensed, Seattle, WA, USA: Lean Kanban University, 2016.
- [16] Atlassian, «Gitflow Workflow,» [En línea]. Available: <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>. [Último acceso: Diciembre de 2018].
- [17] T. Preston-Werner, «Semantic Versioning 2.0.0,» [En línea]. Available: <https://semver.org/>. [Último acceso: Diciembre de 2018].
- [18] H. Maimbo y G. Pervan, Designing a Case Study Protocol for Application in IS research, PACIS, 2005.
- [19] Xebialabs, «xebialabs.com,» [En línea]. Available: <https://xebialabs.com/the-ultimate-devops-tool-chest/repository-management/>. [Último acceso: Diciembre de 2018].
- [20] Stackify, «stackify.com,» 10 3 2017. [En línea]. Available: <https://stackify.com/top-devops-tools/>. [Último acceso: Diciembre de 2018].
- [21] Wikipedia, «Programas de Control de Versiones,» 2 3 2017. [En línea]. Available: https://es.wikipedia.org/wiki/Programas_para_control_de_versiones. [Último acceso: Diciembre de 2018].
- [22] Wikipedia, «Comparison of continuous integration software,» 4 12 2018. [En línea]. Available: https://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software. [Último acceso: Diciembre de 2018].
- [23] Wikipedia, «List of build automation software,» 4 10 2018. [En línea]. Available: https://en.wikipedia.org/wiki/List_of_build_automation_software. [Último acceso: Diciembre de 2018].

[24] Wikipedia, «Operating system level virtualization,» 28 12 2018. [En línea]. Available: https://en.wikipedia.org/wiki/Operating-system-level_virtualization. [Último acceso: Diciembre de 2018].

[25] Wikipedia, «Comparison of issue-tracking systems,» 3 12 2018. [En línea]. Available: https://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems. [Último acceso: Diciembre de 2018].